

# Journal of WSCG

*An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.*

**EDITOR – IN – CHIEF**

**Václav Skala**

***Journal of WSCG***

Editor-in-Chief: Vaclav Skala  
c/o University of West Bohemia  
Faculty of Applied Sciences  
Univerzitni 8  
CZ 306 14 Plzen  
Czech Republic  
<http://www.VaclavSkala.eu> <http://www.wscg.eu>

Managing Editor: Vaclav Skala

Printed and Published by:  
Vaclav Skala - Union Agency  
Na Mazinach 9  
CZ 322 00 Plzen  
Czech Republic

Hardcopy: **ISSN 1213 – 6972**  
CD ROM: **ISSN 1213 – 6980**  
On-line: **ISSN 1213 – 6964**



# Journal of WSCG

## Editor-in-Chief

**Vaclav Skala**

c/o University of West Bohemia  
Faculty of Applied Sciences  
Department of Computer Science and Engineering  
Univerzitni 8  
CZ 306 14 Plzen  
Czech Republic

<http://www.VaclavSkala.eu>

Journal of WSCG URLs: <http://www.wscg.eu> or <http://wscg.zcu.cz/jwscg>

## Editorial Advisory Board MEMBERS

Baranoski,G. (Canada)  
Bartz,D. (Germany)  
Benes,B. (United States)  
Biri,V. (France)  
Bouatouch,K. (France)  
Coquillart,S. (France)  
Csebfalvi,B. (Hungary)  
Cunningham,S. (United States)  
Davis,L. (United States)  
Debelov,V. (Russia)  
Deussen,O. (Germany)  
Ferguson,S. (United Kingdom)  
Goebel,M. (Germany)  
Groeller,E. (Austria)  
Chen,M. (United Kingdom)  
Chrysanthou,Y. (Cyprus)  
Jansen,F. (The Netherlands)  
Jorge,J. (Portugal)  
Klosowski,J. (United States)  
Lee,T. (Taiwan)  
Magnor,M. (Germany)  
Myszkowski,K. (Germany)

Oliveira,Manuel M. (Brazil)  
Pasko,A. (United Kingdom)  
Peroche,B. (France)  
Puppo,E. (Italy)  
Purgathofer,W. (Austria)  
Rokita,P. (Poland)  
Rosenhahn,B. (Germany)  
Rossignac,J. (United States)  
Rudomin,I. (Mexico)  
Sbert,M. (Spain)  
Shamir,A. (Israel)  
Schumann,H. (Germany)  
Teschner,M. (Germany)  
Theoharis,T. (Greece)  
Triantafyllidis,G. (Greece)  
Veltkamp,R. (Netherlands)  
Weiskopf,D. (Canada)  
Weiss,G. (Germany)  
Wu,S. (Brazil)  
Zara,J. (Czech Republic)  
Zemcik,P. (Czech Republic)



# WSCG 2013

## Board of Reviewers

Agathos, Alexander	Fuenfzig, Christoph	Kurt, Murat
Assarsson, Ulf	Gain, James	Kyratzi, Sofia
Ayala, Dolors	Galo, Mauricio	Larboulette, Caroline
Backfrieder, Werner	Gobron, Stephane	Lee, Jong Kwan Jake
Barbosa, Joao	Grau, Sergi	Liu, Damon Shing-Min
Barthe, Loic	Gudukbay, Ugur	Lopes, Adriano
Battiato, Sebastiano	Guthe, Michael	Loscos, Celine
Benes, Bedrich	Hansford, Dianne	Lutteroth, Christof
Benger, Werner	Haro, Antonio	Maciel, Anderson
Bilbao, Javier,J.	Hasler, Nils	Mandl, Thomas
Biri, Venceslas	Hast, Anders	Manzke, Michael
Birra, Fernando	Hernandez, Benjamin	Marras, Stefano
Bittner, Jiri	Hernandez, Ruben Jesus Garcia	Masia, Belen
Bosch, Carles	Herout, Adam	Masood, Syed Zain
Bourdin, Jean-Jacques	Herrera, Tomas Lay	Max, Nelson
Brun, Anders	Hicks, Yulia	Melendez, Francho
Bruni, Vittoria	Hildenbrand, Dietmar	Meng, Weiliang
Buehler, Katja	Hinkenjann, Andre	Mestre, Daniel,R.
Bulo, Samuel Rota	Chaine, Raphaelle	Metodiev, Nikolay Metodiev
Cakmak, Hueseyin	Choi, Sunghee	Meyer, Alexandre
Camahort, Emilio	Chover, Miguel	Molina Masso, Jose Pascual
Casciola, Giulio	Chrysanthou, Yiorgos	Molla, Ramon
Cline, David	Chuang, Yung-Yu	Montrucchio, Bartolomeo
Coquillart, Sabine	Iglesias, Jose,A.	Morigi, Serena
Cosker, Darren	Ihrke, Ivo	Muller, Heinrich
Daniel, Marc	Iwasaki, Kei	Munoz, Adolfo
Daniels, Karen	Jato, Oliver	Murtagh, Fionn
de Geus, Klaus	Jeschke, Stefan	Okabe, Makoto
de Oliveira Neto, Manuel	Jones, Mark	Oyarzun, Cristina Laura
Menezes	Juan, M.-Carmen	Pan, Rongjiang
Debelov, Victor	Kämpe, Viktor	Papaioannou, Georgios
Drechsler, Klaus	Kanai, Takashi	Paquette, Eric
Durikovic, Roman	Kellomaki, Timo	Pasko, Galina
Eisemann, Martin	Kim, H.	Patane, Giuseppe
Erbacher, Robert	Klosowski, James	Patow, Gustavo
Feito, Francisco	Kolcun, Alexej	Pedrini, Helio
Ferguson, Stuart	Krivanek, Jaroslav	Pereira, Joao Madeiras
Fernandes, Antonio	Kurillo, Gregorij	Peters, Jorg

Pina, Jose Luis  
Platis, Nikos  
Post, Frits,H.  
Puig, Anna  
Rafferty, Karen  
Renaud, Christophe  
Reshetouski, Ilya  
Reshetov, Alexander  
Ribardiere, Mickael  
Ribeiro, Roberto  
Richardson, John  
Rojas-Sola, Jose Ignacio  
Rokita, Przemyslaw  
Rudomin, Isaac  
Sacco, Marco  
Salveti, Ovidio  
Sanna, Andrea  
Santos, Luis Paulo  
Sapidis, Nickolas,S.  
Savchenko, Vladimir  
Seipel, Stefan  
Sellent, Anita

Shesh, Amit  
Sik-Lanyi, Cecilia  
Sintorn, Erik  
Skala, Vaclav  
Slavik, Pavel  
Sochor, Jiri  
Sourin, Alexei  
Sousa, A.Augusto  
Sramek, Milos  
Stroud, Ian  
Subsol, Gerard  
Sundstedt, Veronica  
Szecsi, Laszlo  
Teschner, Matthias  
Theussl, Thomas  
Tian, Feng  
Tokuta, Alade  
Torrens, Francisco  
Trapp, Matthias  
Tytkowski, Krzysztof  
Umlauf, Georg  
Vasa, Libor

Vergeest, Joris  
Vitulano, Domenico  
Vosinakis, Spyros  
Walczak, Krzysztof  
WAN, Liang  
Wu, Shin-Ting  
Wuensche, Burkhard,C.  
Wuethrich, Charles  
Xin, Shi-Qing  
Xu, Dongrong  
Yoshizawa, Shin  
Yue, Yonghao  
Zalik, Borut  
Zara, Jiri  
Zemcik, Pavel  
Zhang, Xinyu  
Zhao, Qiang  
Zheng, Youyi  
Zitova, Barbara  
Zwettler, Gerald

# Journal of WSCG

## Vol.21, No.1

### Contents

	Pages
Livesu,M., Scateni,R.: Rigid registration of different poses of animated shapes	1
Atariah,D., Rote,G.: On the Parameterization and the Geometry of the Configuration Space of a Single Planar Robot	11
Hast,A., Nysjö,J., Marchetti,A.: Optimal RANSAC - Towards a Repeatable Algorithm for Finding the Optimal Set	21
Wiesenhuetter,D., Klein,A, Nischwitz,A.: LightCluster - Clustering Lights to Accelerate Shadow Computation	31
Gazolli,K., Salles,E.: Using holistic features for scene classification by combining classifiers	41
Molchanov,V., Fofonov,A., Linsen,L.: Frequency-based Progressive Rendering of Continuous Scatterplots	49
Pennisi,A., Bloisi,D., Gaz,C., Iocchi,L., Nardi,D.: Novel Patterns and Methods for Zooming Camera Calibration	59
Bader,J., Paetzold,M., Kolb,A.: Constrained Up-Scaling for Direct and Global Image Components	69
Oliveira,J. F., Ziebart,M., Iliffe,J., Turner,J., Robson,S.: Trixel Buffer Logic for I/O bound point in N-polygon inclusion tests of massive bathymetric data	79
Trapp,M., Döllner,J.: 2.5D Clip-Surfaces for Technical Visualization	89



# Rigid registration of different poses of animated shapes

Marco Livesu  
Dipartimento di  
Matematica e Informatica  
University of Cagliari  
via Ospedale, 72  
09124 – Cagliari, Italy  
marco.livesu@unica.it

Riccardo Scateni  
Dipartimento di  
Matematica e Informatica  
University of Cagliari  
via Ospedale, 72  
09124 – Cagliari, Italy  
riccardo@unica.it

## ABSTRACT

Different poses of 3D models are very often given in different positions and orientations in space. Since most of the computer graphics algorithms do not satisfy geometric invariance, it is very important to bring shapes into a canonical coordinate frame before any processing. In this paper we consider the problem of finding the best alignment between two or more different poses of the same object represented by triangle meshes sharing the same connectivity. Firstly, we developed a method to select a region of interest (ROI) which has a perfect alignment over the two poses (up to a rigid movement). Secondary, we solved a simplified version of the Largest Common Point-set (LCP) problem with a-priori knowledge about point correspondence, in order to align the ROIs. We eventually align the poses performing least square rigid registration. Our method makes no assumption about the starting positions of the objects and can also be used with more than two poses at once. It is fast, non-iterative, easy to reproduce and brings the poses into the best alignment whatever the initial positions are.

## Keywords

Pose registration; Mesh alignment; Numerical methods.

## 1 INTRODUCTION

Objects are often given in arbitrary position, orientation and scale in space. *Registration* is the process of finding the geometric transformation which brings different sets of data into a congruent coordinate system.

Registration find its application in several fields, like medicine, where data acquired with different modalities (CT, MRI) have to be aligned for joint analysis [31] [33] [20]; image mosaicking [24], where more images have to be merged seamlessly to produce a single, wider, image; creation of super-resolution images [9], where many different images of the same scene contribute to compose a super-resolution representation; computer vision [23], visualization [8], segmentation [10], object recognition, shape matching and retrieval [12], and so on.

In this work we focus our attention on a particular aspect of registration: our goal consists in finding the best alignment between two triangle meshes having the same connectivity and representing the same object in different poses. This is quite important for algorithms that deal with multiple poses, because usually they need them to be aligned. For example, in [25] Marras and colleagues considered a set of poses of the same object as seen from different points of view, in order to find the rigid parts of it and calculate its *motion based segmentation*. As all the computations rely on the silhouettes



Figure 1: An example of pose registration of two different armadillos, achieved using our method. In the inset the ROI used for registering.

the alignment of the shapes is fundamental to properly catch the parts involved in the movement and discard the static ones. This is exactly the kind of algorithm we are targeting: it uses multiple poses, it assumes that the

connectivity is invariant over them and it requires them to be aligned.

Registration is also a fundamental building block of many shape interpolation techniques. One can think for example, given two poses, at the problem of generating the complete sequence which describes the movement bringing from one pose to the other [32]; or, given a set of simple poses, at the problem of summarizing them into a more articulated one [32]. Depending on the algorithms used, for all these settings, a pre-alignment could be either mandatory or suggested. Once again, it is quite common in this scenario to have the same connectivity among the models, because probably they have been generated by the same reference triangle mesh.

The main question to answer to is: when are two poses well aligned? Intuitively a good alignment scheme should take into account only the parts of the object which are not involved in the movement suggested by the considered poses; leaving aside all the parts for which, because of the change of pose, a perfect alignment does not exist any longer. This is definitely true, but not sufficient. Think for example at the horses in Figure 2: there are more patches along the surface which remain unchanged over the two poses. Moreover, it is clear from that image, that such patches are likely to ask for a different rigid registration scheme each other.

How can we handle this? One additional consideration must be done. Our final idea about pose registration is the following: *a good rigid pose registration scheme should take into account, among all the patches which are common in the considered poses, only the patch covering the largest area.*

Indeed, there are a lot of good registration algorithms in literature, but none of them is designed to face this particular problem in which we need to align only a subset of primitives and point correspondences are known a priori. Global methods, like the ICP algorithms provided by Besl and McCay in [11] and by Chen and Medioni in [13] are not suitable to accomplish this task; first of all because they are *global*, that is, they use all the available data to estimate the alignment scheme; secondary, because ICP algorithms always converges monotonically to the nearest local minimum of a mean square distance metric. This means that a coarse pre-alignment is mandatory to avoid to fall in the wrong local minima.

In recent years many other registration algorithms have been proposed to the scientific community. In [4] Aiger and colleagues proposed a randomized alignment scheme which is based on approximately coplanar 4-points sets. Their algorithm is fast and resilient to noise. However, we made some tests with the implementation provided in [2], and we found that the results were not satisfactory. The motivation is that such algorithm has

been designed to align range maps, which usually does not have a priori correspondences and also require a high percentage of overlapping, say more than 40%. Since poses could be strongly different each other, we need algorithms able to perform well, even in presence of lower overlapping percentages.

Other algorithms are based on global quantities, such as Principal Component Analysis or centroids (see for example the method proposed by Chaouch and Verroust-Blondet in [12]). They are not suitable too. The reason is that a global descriptor computed over a shape bears an arbitrary relationship to the value that would be computed for a different pose of the same shape. It follows that is impossible to align poses employing this kind of techniques.

## 1.1 Main contribution

Our **main contribution** to solving the problem of rigid pose registration is that we provide a simple method to localize the **greatest surface patch which preserve its appearance over the poses**. Such method is based on the discrete Gaussian curvature, which is one of the most familiar of all local shape descriptors (to have another example of application of the Gaussian curvature one can refer to [19]). The pose registration is eventually achieved by applying a state of the art least square rigid registration algorithm.

The rest of the paper is organized as follows: in Section 2 we briefly recap the mathematical notions over which we built our work. In Section 3 we summarize our proposal. In Sections 4 and 5 we discuss all the technical details of the core of our algorithm. In Section 6 we analyze three different approaches to multiple pose registration. In Section 7 we present and discuss the results we have obtained. In Section 8 we discuss on the limitations our method have, and, finally, in Section 9 we draw the conclusions.

## 2 MATHEMATICAL BACKGROUND

In this section we summarize the basic notions useful to better understand our approach. After a brief introduction to the differential geometry of surfaces, focusing in particular on discrete Gaussian curvature, we introduce the Largest Common Point-set (LCP) problem along with its most popular metrics.

### 2.1 Curvature in Differential Geometry

Let  $S$  be a  $\mathcal{C}^\infty$  surface embedded in  $\mathbb{R}^3$ : curvatures describe the local bending of the surface. For every point  $s \in S$  the two *principal curvatures*  $k_1$  and  $k_2$  are respectively the maximum and the minimum normal curvatures, measured in their orthogonal *principal directions*  $\mathbf{e}_1$  and  $\mathbf{e}_2$ . The *mean curvature* can then be defined as



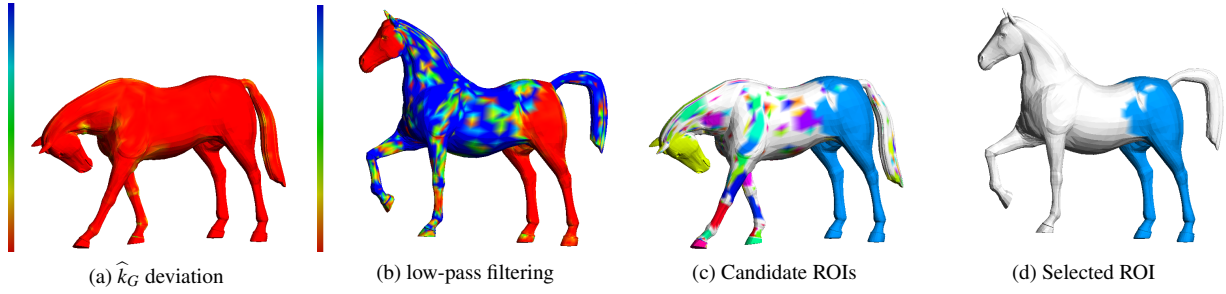


Figure 2: Region of interest selection. From left to right: Gaussian curvature displacements between the considered poses; Gaussian curvature displacements after the cutting-off of the 20% most affected vertices; all the candidate ROIs and, finally, the ROI selected to align the poses.

$k_H = (k_1 + k_2) / 2$ . The *Gaussian curvature*  $k_G$ , instead, is defined as the product of the two principle curvatures  $k_G = k_1 \cdot k_2$ . Mean and Gaussian curvatures are among the most important local properties of a surface [16] [34].

Moving from  $\mathcal{C}^\infty$  surfaces to meshes (which one can view as  $\mathcal{C}^0$  surfaces) the definitions provided above need to be reformulated. In discrete differential geometry the geometric properties of the surface at each vertex are considered as spatial averages around this vertex. Such average is usually restricted to the triangles incident to the vertex itself, which is often referred as the 1-ring or star neighborhood.

Meyer and colleagues [26] provide a discretization of the Gaussian curvature formula; the derived pointwise discrete Gaussian curvature operator is

$$\hat{k}_G(\mathbf{x}_i) = \frac{1}{\mathcal{A}} \left( 2\pi - \sum_{j=1}^{\#f} \phi_j \right)$$

where  $\phi_j$  is the angle of the  $j$ -th face at the vertex  $\mathbf{x}_i$ , and  $\#f$  denotes the number of faces around this vertex.

As we previously said, the curvature over a mesh vertex is calculated as the spatial average around the vertex itself. The definition of area we use is known as *Voronoi area* and, for a vertex  $\mathbf{x}_i$ , is defined in function of its neighbors  $\mathbf{x}_j$  as

$$\mathcal{A}_{\text{Voronoi}} = \frac{1}{8} \sum_{j \in N(\mathbf{x}_i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \|\mathbf{x}_i - \mathbf{x}_j\|,$$

where  $\alpha_{ij}$  and  $\beta_{ij}$  are the angles measured in the opposite corners with respect to the edge joining  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . This area proved to be sufficient in all our experiments. However, our approach does not depend on the particular choice of  $\mathcal{A}$ . Mixed area, for instance, would accommodate obtuse triangles and its application is straightforward [26].

## 2.2 The LCP problem

The problem of finding the largest set of points which is congruent to a subset of each input data is well known in literature under the name of *Largest Common Point-set problem* (LCP) [6] [7]. It has been studied both from the theoretical and practical points of view and it is widely used in different fields like computational biology and chemistry (there is a frequent need to extract a common pattern from multiple data), and surface reconstruction from 3D scans (different range maps partially overlapped have to be aligned before extracting the final surface [4]).

Given two point sets  $A$  and  $B$ , the LCP between  $A$  and  $B$  is the maximal subset  $A' \subseteq A$  which is geometrically congruent to some subset  $B'$  of  $B$ . This implies that two subsets are said to match only when the underlying geometric transformation takes each point of  $A'$  exactly onto one point of  $B'$ . This statement is also known as largest common point set problem with *exact matching metric* [22] [27] [5].

In real applications, congruence between subsets is a restriction too tight, a similarity metric ( $\epsilon$ -congruence) is used instead. Two common metrics for quantifying the notion of similarity are the *Hausdorff distance* [14] [15] [21], and the *bottleneck matching metric* [17].

The former is defined as the maximum distance between a point in one set and its nearest neighbor in the other set; the latter seeks a perfect bipartite matching between two equal cardinality point sets such that the maximum distance between any two matched points is minimized.

It is worth noticing that most of the problems we want to solve with the LCP scheme, especially those involving three and higher dimensional point sets, demand a one-to-one matching between two point sets, making the Hausdorff metric ill-posed. This motivates the study of the problem using the bottleneck matching metric.

## 3 OVERVIEW

Our method can align pairs of meshes representing different poses of the same object. This implies that the

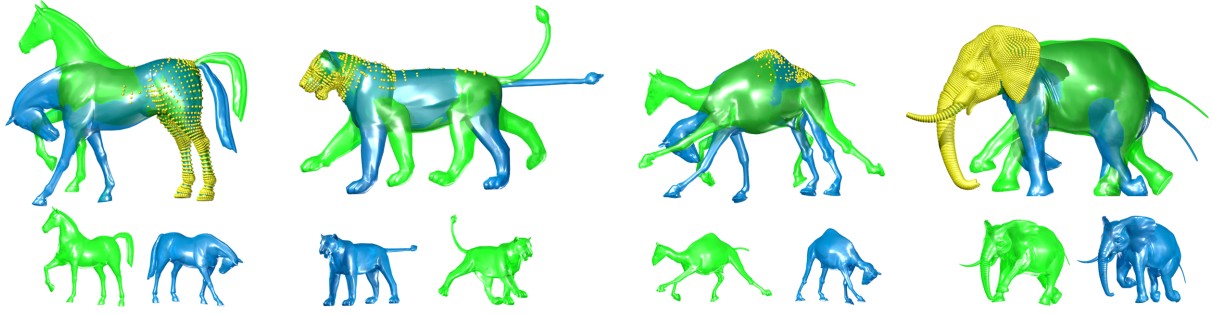


Figure 3: From left to right: mesh registration between two horses, two lions, two camels and two elephants. In the bottom row there are the meshes in the original position and orientation, before registration.

cardinality of the vertices vector is the same, and the two meshes share an identical connectivity.

Let  $\mathcal{M} = (V, K)$  and  $\mathcal{M}' = (V', K)$  be the meshes we want to align, where  $K$  describes the connectivity and  $V = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  and  $V' = \{\mathbf{v}'_1, \dots, \mathbf{v}'_n\}$  describe the geometric positions of the vertices in  $\mathbb{R}^3$ . In our approach we take advantage of the particular nature of the problem, setting up a simplified version of the largest common point set problem (LCP) with bottleneck matching metric and a priori knowledge about point correspondences between the vertices of the two shapes. In this simplified framework, once defined the areas we want to overlap, we only have to solve for the isometry which performs such alignment.

We can summarize the algorithm in four steps:

1. Calculation of the Gaussian curvature for each vertex of  $\mathcal{M}$  and  $\mathcal{M}'$ ;
2. Selection of the subset of vertices we want to overlap among those having similar Gaussian curvature in both poses (see Section 4);
3. Estimation of the rigid movement which makes  $\mathcal{M}$  and  $\mathcal{M}'$   $\varepsilon$ -congruent (see Section 5);
4. Pose registration achievement by applying the geometric transformation obtained at the previous step.

#### 4 REGION OF INTEREST (ROI)

Let  $V_{k_G} \subseteq V$  be the subset of vertices of  $\mathcal{M}$  having similar Gaussian curvature in both meshes, up to a tolerance threshold  $\varepsilon$

$$V_{k_G} = \left\{ \mathbf{v}_i \in V \mid \left| \hat{\kappa}_G(\mathbf{v}_i) - \hat{\kappa}_G(\mathbf{v}'_i) \right| \leq \varepsilon, \mathbf{v}'_i \in V' \right\}.$$

The distribution of the points in  $V_{k_G} \subseteq V$  among the surface identifies which parts of the surface have common behaviour in both poses. Starting from this points we

perform region growing in order to define a set of candidate regions of interest for the alignment. Let  $\mathbf{v}_i$  be a vertex in a candidate ROI and let  $N(i)$  be the set of vertices sharing an edge with it (i.e., its one ring). If the average Gaussian curvature deviation

$$\overline{\Delta \kappa_G}(\mathbf{v}_i) = \frac{\sum_{j \in N(i)} |\hat{\kappa}_G(\mathbf{v}_i) - \hat{\kappa}_G(\mathbf{v}'_j)|}{\#N(i)}$$

is lower than a predefined threshold  $\sigma$ , the candidate ROI expands over  $N(i)$ . We repeated this process iteratively starting from each seed point until convergence. In Figure 2c the candidate ROIs for two poses of the horse model.

The ROI selection depends on the threshold  $\sigma$ . As one can note from Figure 2a, curvature deviation is ill-distributed over the surface: few vertices have a huge displacement, thus flattening the rest of the distribution. In this scenario the automatic research of a good  $\sigma$  value can be very difficult. We observed that vertices with high curvature deviation are located in the areas involved in the change of pose and, thus, they are meaningless for the registration problem. We therefore decided to restrict the research to a subset of vertices with small curvature deviation. Let  $\mathcal{L}\mathcal{D}$  be the indexes of the vertices less affected by curvature deviation (80% of the total number of vertices); we eventually compute the  $\sigma$  value as

$$\sigma = 0.2 \arg \max_{i \in \mathcal{L}\mathcal{D}} \left| \hat{\kappa}_G(\mathbf{v}_i) - \hat{\kappa}_G(\mathbf{v}'_i) \right|.$$

Cutting-off of the vertices with greater deviation makes the error distribution easier to be treated algorithmically (see Figure 2b). All the results shown in this paper have been achieved using the automatic  $\sigma$  calculation described above.

We have now a problem with selecting the best ROI to guide the whole registration process. Our idea is to consider only the widest patch, thus we have to estimate the area of each of them. The discrete Gaussian curvature measured in a vertex of a mesh is the average of the Gaussian curvatures computed in the area  $\mathcal{A}$  around it

(see section 2). This area becomes now twice important: first because it is involved in the discrete  $\hat{\kappa}_G$  formula; second because also gives the possibility to measure the global area of each candidate ROI as the sum of the areas associated to each vertex in it. This is quite important because makes the measure of the patch's size independent from the underlying triangulation we are dealing with.

## 5 ALIGNMENT

Given a region of interest  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$  in the first pose, we want to find the rigid movement that optimally aligns  $\mathcal{P}$  and its dual in the second pose  $\mathcal{P}'$ . The transformation  $T : \mathcal{P} \rightarrow \mathcal{P}'$  we are looking for has the form

$$T = \begin{pmatrix} R & \mathbf{t} \\ 0^T & 1 \end{pmatrix},$$

where  $R \in \mathbb{R}^{3 \times 3}$  is a pure rotation matrix (i.e.  $R^T R = I$ ) and  $\mathbf{t} \in \mathbb{R}^3$  is a translation. If  $\mathcal{P}$  and  $\mathcal{P}'$  were strictly congruent, we would have  $\|\mathbf{p}'_i - T(\mathbf{p}_i)\| = 0$ ,  $i = 1, \dots, n$ . Unfortunately, congruence is very unlikely to happen; in the common scenario a perfect alignment does not exist, therefore we should find the best approximating rotation and translation that fit  $\mathcal{P}$  and  $\mathcal{P}'$  in the least square sense, i.e. minimizes

$$(\tilde{R}, \tilde{\mathbf{t}}) = \arg \min_{R, \mathbf{t}} \sum_{i=1}^n \|(R\mathbf{p}_i + \mathbf{t}) - \mathbf{p}'_i\|^2. \quad (1)$$

This is a very well known problem in literature and there are several existing algorithms to solve it. A comparison between four of them can be found in [18] while a weighted instance of the problem has been studied by Sorkine and Alexa in [29].

The best translation can be found by taking the derivative of (1) w.r.t.  $\mathbf{t}$  and searching for its roots. One can eventually find that

$$\tilde{\mathbf{t}} = \bar{\mathbf{p}}' - R\bar{\mathbf{p}},$$

with

$$\bar{\mathbf{p}} = \frac{1}{n} \sum_{i=1}^n \mathbf{p}_i \quad \bar{\mathbf{p}}' = \frac{1}{n} \sum_{i=1}^n \mathbf{p}'_i.$$

In other words the optimal translation  $\tilde{\mathbf{t}}$  maps the transformed centroid of  $\mathcal{P}$  in the centroid of its dual  $\mathcal{P}'$ . To find the best rotation we can now consider the centered vectors

$$\mathbf{x}_i = \mathbf{p}_i - \bar{\mathbf{p}} \quad \mathbf{y}_i = \mathbf{p}'_i - \bar{\mathbf{p}}'$$

$i = 1, \dots, n$ , and the Singular Value Decomposition (SVD) of their  $3 \times 3$  covariance matrix  $XY^T$  ( $X$  and

$Y$  are two  $3 \times n$  matrices containing all the centered vectors ordered by column)

$$XY^T = U\Sigma V^T$$

The rotation we are looking for is

$$R = V \begin{pmatrix} 1 & & \\ & 1 & \\ & & \det(VU^T) \end{pmatrix} U^T.$$

The last term in the diagonal matrix is setted to  $\det(VU^T)$  instead of 1 in order to avoid reflections. A formal and clear demonstration of the whole minimization process can be found in [28].

## 6 REGISTRATION OF MULTIPLE POSES

Our method can be also used to align more than two poses at once. Given a set of poses  $\mathcal{M}_1, \dots, \mathcal{M}_n$ , three different strategies are possible:

1. if we need to select a *reference pose*  $\mathcal{M}_{\text{ref}}$ , it can be used to align all the other poses;
2. if there is a time-sequence leading to changes in the poses, they can be registered in chain, that is,  $\mathcal{M}_1$  vs  $\mathcal{M}_2$ ,  $\mathcal{M}_2$  vs  $\mathcal{M}_3$ ,  $\dots$ ,  $\mathcal{M}_{n-1}$  vs  $\mathcal{M}_n$ ;
3. if no reference pose and time-sequence exists, the poses can be registered all together.

What is the best multi-registration schema? Actually, each strategy has pros and cons: the best choice depends on the applications. To summarize multiple simple poses into an articulated one, the first scheme is likely to be the best one. To add frames at a discrete sequence of poses describing a movement, the second scheme would be better.

## 7 RESULTS AND DISCUSSION

We implemented our methods in C++, using the VCG Library [3] for the manipulation of geometric data structures, and the Eigen library [1] for the numerical computations. Experiments were run on a iMac equipped with 2.66GHz Intel Core 2 Duo and 4GB RAM, using a single core. We used as dataset the shapes provided by Sumner and Popović in their deformation transfer for triangle meshes's web page [30]. For each shape we considered many different poses: from a minimum of 8 poses (horse), to a maximum of 47 poses (elephant gallop).

Our tests have been organized as follows: for each couple of poses we run our algorithm three times, each time applying a random rigid movement to the shapes, in

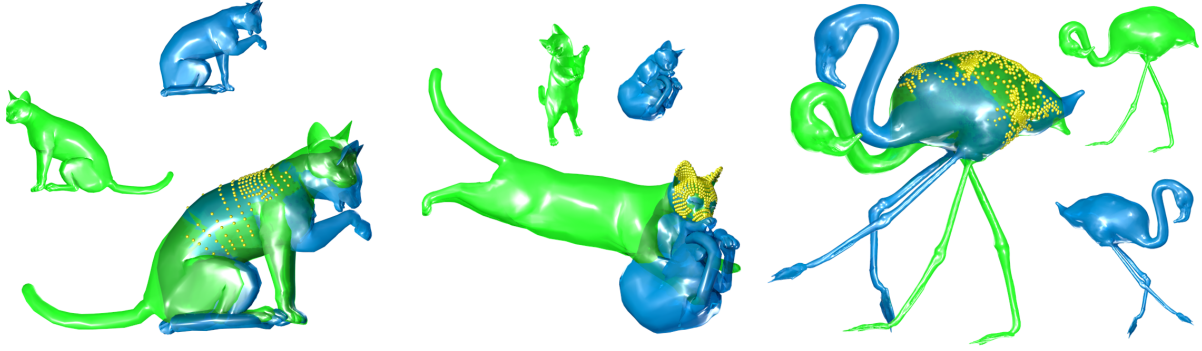


Figure 4: Some examples of registration achieved with our algorithm. In yellow, the subsets of mesh vertices used to align the poses.

order to put them in general position. We had been able to find a good alignment in every test we made (see Figures 3 and 4). In particular, for each couple of poses, we always get the same numerical results, emphasizing the robustness of our shape descriptor (i.e. discrete Gaussian curvature displacements) with respect to isometries. In Table 1 we report timings for each registration shown in this paper. One can note that, since the most time consuming task is the registration step, the complexity of the whole algorithm depends on the size of the ROI rather than the complexity of the mesh (i.e. the number of vertices).

### 7.1 Multiple poses

For our experiments, we used the horse gallop sequence dataset. Both reference pose and time sequence are available for it, so we had been able to test each possible strategy.

Theoretically, the third method presented in Section 6 is more computationally expensive than the first two: it is intrinsically quadratic, while the others are linear. But, what we actually do in the third method, is the registration of all the poses versus a reference one using a ROI computed on all the meshes. This leads to timing that are comparable each other. As we can see in Table 2, all the methods require the same time for the computation of the curvature and alignment and the last one (all vs all), is definitely the fastest for ROI computation (due to the smaller number of vertices in the ROI). An example of multiple registration is given in Figure 5.

We studied the quality of the registration schemas, measuring the Mean Square Error (MSE) metric over the vertices of the ROI:

$$MSE(\mathcal{M}_a, \mathcal{M}_b) = \frac{\sum_{i \in \text{ROI}_{ab}} \|v_i^{(a)} - v_j^{(b)}\|_2^2}{\#\text{ROI}_{ab}}.$$

We computed MSE errors over each possible couple of poses, in order to have a reference ROI and a reference error measure. We, then, tested each multi-registration

strategy, and measured the registration errors over the, previously determined, reference ROI.

For each pair of horse meshes that we tested, MSE error was very close to zero (Figure 6a), emphasizing the registration capabilities of the method. Multiple pose registration over a reference pose behaves well, however some error peaks between two particular non-reference poses can occur, for example between the third and the eighth horses (Figure 6b). Chain registration suffers the same problem: error is low for most of the poses but there are peaks between first and last poses of the chain (Figure 6c). Finally, as one could expect, we found that the best error distribution over the poses have been achieved considering all the poses together (Figure 6d).

## 8 LIMITATIONS

The proposed algorithm performs well with all the shapes we tested on. It has, however, some limitations, in the sense that there are no guarantees that the registration provided will be *good* for an end user, in other words, the one the user expected. In particular, if the model contains more parts that move rigidly and cover more or less the same area the choice between these parts as candidate ROI will be unstable. Moreover, in case the selected ROI lies in a peripheral area of the shape (e.g., a foot, a head or a hand) the result could be unnatural. The definition of *natural registration* is



Figure 5: Multiple registrations of eight poses of the Horse mesh.

Model	Vertices	ROI	Curvature (ms)	ROI (ms)	Alignment (ms)	Total (ms)
Armadillos	165,954	3,541	307	126	6	439
Elephants	42,321	19,339	189	98	28	315
Flamingos	26,907	1,092	120	44	10	174
Camels	21,887	376	72	34	25	131
Horses	8,431	2,491	35	27	11	73
Cats 1	7,207	965	32	23	4	59
Cats 2	7,207	544	32	22	2	56
Lions	5,000	1,751	24	15	8	47
Cactus	5,261	410	22	2	3	27

Table 1: Running times of our algorithm, in milliseconds. In the rightmost column there is the total time needed to align the poses while in the previous three columns there are the timings needed, respectively, for Gaussian curvature calculation (Curvature), Region Of Interest determination (ROI), and the least square registration calculation (Alignment). Second and third columns show, for each pair of poses, respectively the number of mesh Vertices (Vertices) and the cardinality of the Region Of Interest (ROI) we used to align.

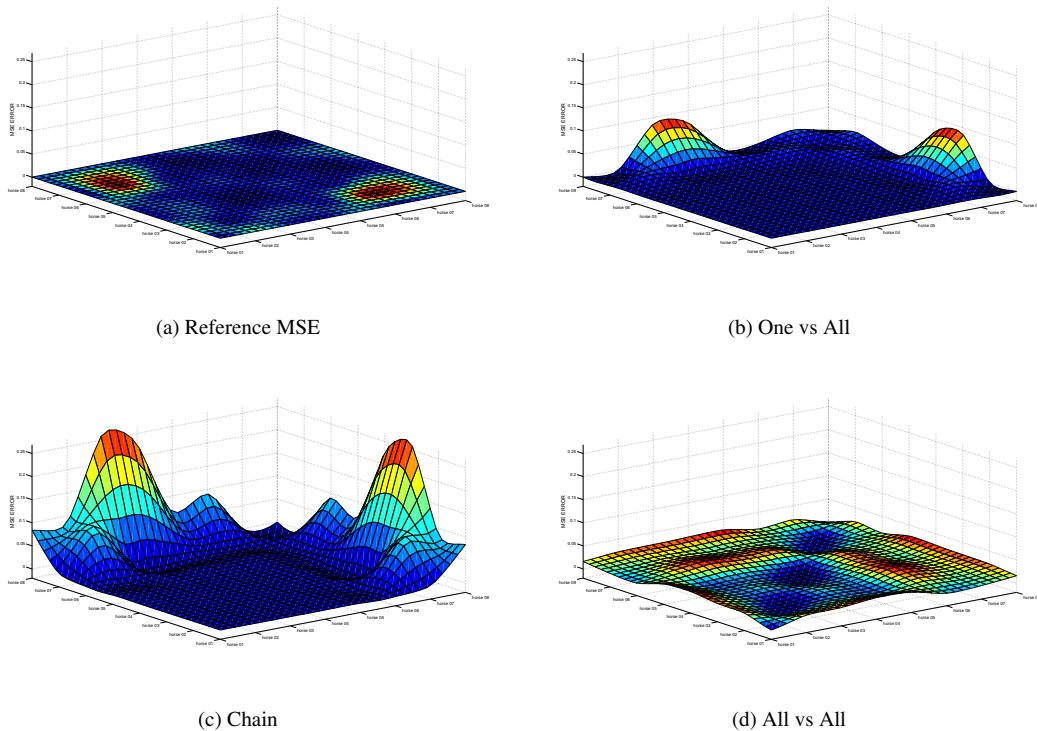


Figure 6: Mean Square Error plots for multiple pose registration. From left to right, up to down: reference MSE error measured between each possible pair of the sequence (a); MSE error obtained aligning all the poses in the sequence against a reference pose (b); MSE error obtained aligning all the poses in chain (c); and MSE error obtained comparing all the poses together (d).



somehow related to the human perception, it is the kind of registration that a human can imagine when looks at two different poses of the same object. We have tried to set up a mathematical model which works like a human should work in the general case but, unfortunately, natural registrations are as easy to do for humans as incredibly difficult for machines, because is a matter of shape understanding and matching.

One second limitation regards noise. The algorithm is noise insensitive as long as all the models involved in the computations are affected by the same kind and the same amount of noise. In any other case, the Gaussian curvature values would be totally unreliable, leading to wrong registrations. However, this is usually not a big deal: in shape interpolation and modelling one tends to use almost the same meshes, deformed and readapted to the new pose, thus bearing the same amount of noise.

## 9 CONCLUSIONS

Pose registration is the process of finding the best possible alignment among different meshes representing the same shape in different positions. Since poses can be strongly different each other the best alignment method should be able to work only with local patches of the surface, that is the regions of the shape having the same properties (mainly curvature) in the given poses.

In this paper we presented a new, non-iterative, pose registration scheme, which employs Gaussian curvature as local shape invariant and we show that the proposed method can be fast, robust and accurate in the most relevant cases of pose registration: interpolation among different poses and modelling.

## 10 ACKNOWLEDGMENTS

The authors would like to thank Robert Sumner and Jovan Popović for making available on their website their 3D shapes, which were useful for our work. We would like also to thank our colleagues, Marianna Saba, Enrico Puppo and Daniele Panozzo, for the fruitful discussions had during this work.

## 11 REFERENCES

[1] <http://eigen.tuxfamily.org/>

Method	Curvature (ms)	ROI (ms)	Alignment (ms)	Total (ms)
one vs all	160	209	46	415
chain	158	190	55	403
all vs all	160	53	72	285

Table 2: Comparison of the the three multiple poses registration methods.

- [2] <http://meshlab.sourceforge.net/>
- [3] <http://vcg.sourceforge.net/>
- [4] Aiger, D., Mitra, N.J., Cohen-Or, D.: 4-points congruent sets for robust pairwise surface registration. In: ACM SIGGRAPH 2008 papers, SIGGRAPH '08, pp. 85:1–85:10. ACM, New York, NY, USA (2008)
- [5] Akutsu, T.: On determining the congruence of point sets in  $d$  dimensions. *Comput. Geom. Theory Appl.* **9**, 247–256 (1998)
- [6] Akutsu, T., Halldórsson, M.M.: On the Approximation of Largest Common Subtrees and Largest Common Point Sets. In: Proceedings of the 5th International Symposium on Algorithms and Computation, ISAAC '94, pp. 405–413. Springer-Verlag, London, UK (1994)
- [7] Akutsu, T., Tamaki, H., Tokuyama, T.: Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets. In: Proceedings of the thirteenth annual symposium on Computational geometry, SCG '97, pp. 314–323. ACM, New York, NY, USA (1997)
- [8] Allen, P., Feiner, S., Troccoli, A., Benko, H., Ishak, E., Smith, B.: Seeing into the Past: Creating a 3D Modeling Pipeline for Archaeological Visualization. In: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, 3DPVT '04, pp. 751–758. IEEE Computer Society, Washington, DC, USA (2004)
- [9] Arican, Z., Frossard, P.: Super-resolution from unregistered omnidirectional images. In: International Conference on Pattern Recognition, pp. 1–4 (2008)
- [10] Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A.: Mesh Segmentation - A Comparative Study. In: Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on, p. 7 (2006)
- [11] Besl, P.J., McKay, N.D.: A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 239–256 (1992)
- [12] Chaouch, M., Verroust-Blondet, A.: A novel method for alignment of 3D models. In: Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on, pp. 187–195 (2008)
- [13] Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. *Image Vision Comput.* **10**, 145–155 (1992)
- [14] Chew, L.P., Dor, D., Efrat, A., Kedem, K.: Geometric Pattern Matching in  $d$ -Dimensional Space. *Discrete & Computational Geometry* **21**, 257–274 (1999)

- [15] Chew, L.P., Goodrich, M.T., Huttenlocher, D.P., Kedem, K., Kleinberg, J.M., Kravets, D.: Geometric pattern matching under Euclidean motion. *Comput. Geom. Theory Appl.* **7**, 113–124 (1997)
- [16] Do Carmo, M.P.: *Differential Geometry of Curves and Surfaces*. Prentice Hall (1976)
- [17] Efrat, A., Itai, A., Katz, M.J.: Geometry helps in bottleneck matching and related problems. *Algorithmica* **31**, 1–28 (2001)
- [18] Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl.* **9**, 272–290 (1997)
- [19] Guo, K., Li, M.: A Novel Shape Descriptor: Gaussian Curvature Moment Invariants. In: *Proceedings of the 2009 First IEEE International Conference on Information Science and Engineering, ICISE '09*, pp. 1087–1090. IEEE Computer Society, Washington, DC, USA (2009)
- [20] Hill, D.L.G., Batchelor, P.G., Holden, M., Hawkes, D.J.: Medical image registration. *Physics in Medicine and Biology* **46**(3), R1 (2001)
- [21] Huttenlocher, D.P., Kedem, K., Sharir, M.: The upper envelope of Voronoi surfaces and its applications. In: *Proceedings of the seventh annual symposium on Computational geometry, SCG '91*, pp. 194–203. ACM, New York, NY, USA (1991)
- [22] Irani, S., Raghavan, P.: Combinatorial and experimental results for randomized point matching algorithms. In: *Proceedings of the twelfth annual symposium on Computational geometry, SCG '96*, pp. 68–77. ACM, New York, NY, USA (1996)
- [23] Jin, H., Favaro, P., Soatto, S.: A semi-direct approach to structure from motion. *The Visual Computer* **19**, 377–394 (2003)
- [24] Kouroggi, M., Kurata, T., Hoshino, J., Muraoka, Y.: Real-Time Image Mosaicing from a Video Sequence. In: *International Conference on Image Processing*, pp. 133–137 (1999)
- [25] Marras, S., Bronstein, M.M., Hormann, K., Scateni, R., Scopigno, R.: Motion-based mesh segmentation using augmented silhouettes. *Graphical Models* **74**(4), 164–172 (2012)
- [26] Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In: H.C. Hege, K. Polthier (eds.) *Visualization and Mathematics III*, pp. 35–57. Springer-Verlag, Heidelberg (2003)
- [27] de Rezende, P.J., Lee, D.T.: Point Set pattern matching in d-dimensions. *Algorithmica* **13**, 387–404 (1995)
- [28] Sorkine, O.: Least-Squares Rigid Motion Using SVD. Tech. rep. (2009)
- [29] Sorkine, O., Alexa, M.: As-Rigid-As-Possible Surface Modeling. In: A. Belyaev, M. Garland (eds.) *SGP07: Eurographics Symposium on Geometry Processing*, pp. 109–116. Eurographics Association, Barcelona, Spain (2007)
- [30] Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. *ACM Trans. Graph.* **23**(3), 399–405 (2004)
- [31] Tao, J.X., Hawes-Ebersole, S., Baldwin, M., Shah, S., Erickson, R.K., Ebersole, J.S.: The accuracy and reliability of 3D CT/MRI co-registration in planning epilepsy surgery. *Clinical Neurophysiology* **120**, 748–753 (2009)
- [32] Winkler, T., Drieseborg, J., Alexa, M., Hormann, K.: Multi-Scale Geometry Interpolation. *Computer Graphics Forum* **29**(2), 309–318 (2010)
- [33] Yang, Z., Liu, L., Liu, W., Yang, W.: Research of CT/MRI Tumor Image Registration Based on Least Square Support Vector Machines. In: *International Conference on Intelligent Computing*, pp. 1078–1086 (2008)
- [34] Yamauchi, H., Lee, S., Lee, Y., Ohtake, Y., Belyaev, A., Seidel, H.P.: Feature Sensitive Mesh Segmentation with Mean Shift. In: *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pp. 238–245 (2005)





# On the Parameterization and the Geometry of the Configuration Space of a Single Planar Robot

Dror Atariah

Institut für Informatik  
Freie Universität Berlin  
Berlin, Germany

dror.atariah@fu-berlin.de

Sunayana Ghosh

Institut für Informatik  
Freie Universität Berlin  
Berlin, Germany

sghosh@inf.fu-berlin.de

Günter Rote

Institut für Informatik  
Freie Universität Berlin  
Berlin, Germany

rote@inf.fu-berlin.de

## ABSTRACT

Translating and rotating planar polygonal robots are studied in the literature for decades. An integral part of this study is the configuration space which corresponds to the work space. In the context of motion planning problems, the boundary between the free and forbidden parts of the configuration space plays a major role. In this paper we find an explicit parameterization of the boundary of the forbidden space. Using this parameterization we detail several geometrical properties of the various elements which constitute this boundary. In addition, this parameterization enables us to visualize these elements.

## Keywords

Robotics, Motion planning, Configuration Space, Parameterizations

## 1 INTRODUCTION

The *piano movers problem* is about four decades old [SS83, IKP73] and studied intensively ever since. A fundamental part of this study is the configuration space which is associated to the work space at hand. A work space which consists of a planar polygonal convex robot, which is free to rotate and translate and polygonal obstacles give rise to a *configuration space*. Each point in the configuration space corresponds to a unique placement or pose of the robot in its work space, and vice versa, that is, every pose of the robot in the work space corresponds to a unique configuration point. The presence of obstacles in the work space translates to the partition of the configuration space into two parts, namely the *free* and *forbidden* spaces. Configuration points in the forbidden part correspond to poses in which the interior of the robot intersects the interior of one or more obstacles.

Most studies set the solution of the *motion planning problem* as the primary goal and thus focus mainly on algorithmical aspects. Thus, the related configuration space was hardly studied from a *geometrical point of view*. In this paper we focus on the geometrical properties of the configuration space which is associated to

the piano movers problem. To that end, we derive in Section 3 an explicit parameterization of the boundary of the forbidden space. Better understanding of this boundary can contribute, for example, to the general study of the motion planning problem. In turn, using this parameterization we study in Section 4 the geometrical properties of this boundary.

In terms of visualization, most of the illustrations of the configuration space that can be found in the literature are rather simple. It is well known that for a robot which can *only* translate, the boundary of the forbidden space is polygonal and can be computed using *Minkowski sums*. Thus, most of the visualizations slice the configuration space with horizontal planes. Each slice corresponds to a fixed rotation of the robot and the boundary can be computed using Minkowski sums. Finally, stacking these slices yields a discrete visualization of the obstacles as they appear in the configuration space [Lat93]. Using our parameterization it is easy to visualize the boundary of the forbidden space, as can be seen in Figure 9 and in [AR12].

## Previous Work.

The work of Lozano-Pérez and Wesley [LPW79] put the so-called *configuration space* under the spotlight. Surveys like [WB00, HA92] provide a broad overview at least on the early study of this fundamental concept. As we already pointed out, the literature aims mainly at the motion planning problem and hardly considers the boundary of the forbidden space per se, let alone parameterizing it.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

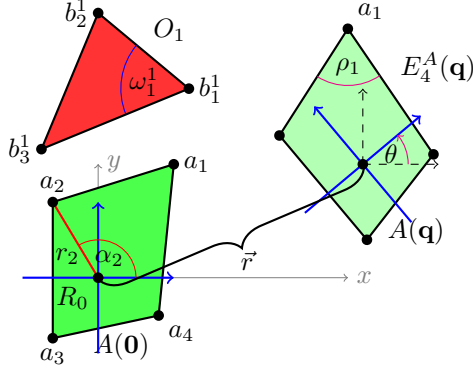


Figure 1: An example of a work space with a robot  $A(\mathbf{0})$  in its rest position (dark green) and in a configuration resulting from a translation by a vector  $\vec{r}$  and rotation in angle  $\theta$  (light green), that is  $A(\mathbf{q})$  where  $\mathbf{q} = (\vec{r}, \theta)$ . Local frame is in solid blue and one obstacle  $O_1$  is in red.

Two interesting examples are [BA88, MST13]. Both attempt to solve the motion planning problem itself, although they provide, as a byproduct, some idea on the geometrical nature of the boundary of the forbidden space. Yet, neither of them provides an explicit, simple and concise representation of the various elements of the boundary.

## 1.1 Definitions and Notations

In this section we describe all the notations and notions that will be used throughout the paper. Further details can be found in standard textbooks like [LaV06, Lat93, CLH<sup>+</sup>05, Lau98]. In addition refer to Figure 1 for illustrations of the various definitions.

### The Robot.

A robot  $A$  is a convex planar polygon with  $n$  vertices denoted by  $\{a_i\}_{i=1}^n$ ; we assume that they are given in counterclockwise order. The robot can translate and rotate in a *work space* scattered with polygonal (convex) obstacles. The work space is denoted by  $\mathcal{W}$  and we take it to be  $\mathcal{W} = \mathbb{R}^2$ . The *reference point* of the robot is denoted by  $R_0$ , and we assume that in the *rest position* it is at the origin. Furthermore, we assume that the *local frame* of the robot aligns with the coordinates of the work space when in the rest position.

The vertices of the robot are either given with Cartesian coordinates, or with polar coordinates. For the vertex  $a_i$  we use the following notation  $a_i = r_i(\cos \alpha_i, \sin \alpha_i)$  where  $r_i = \|a_i\|$  and  $\alpha_i$  denotes the angle with respect to the local frame of the robot. We assume that the vertices are in an increasing angular order, that is  $0 \leq \alpha_1 \leq \dots \leq \alpha_n < 2\pi$ . Finally, for the sake of simplicity and consistency, we assume that the reference point lies in

the interior or on the boundary of the robot. We denote by  $E_i^A$  the edge  $\overline{a_i a_{i+1}}$  which connects  $a_i$  with  $a_{i+1}$ . Lastly, we denote by  $\rho_i$  the internal angle corresponding to the  $i$ -th vertex.

### Obstacle(s).

Let  $\{O_k\}_{k=1}^m$  be  $m$  obstacles in the work space. The *vertices* of  $O_k$  for some  $k$  are given in counterclockwise order, and are denoted by  $\{b_j^k\}$ . Analogously to the notations we use for the robot,  $E_j^{O_k}$  will denote the edge which connects  $b_j^k$  and  $b_{j+1}^k$ . Finally, the interior angle at the  $j$ -th vertex will be denoted by  $\omega_j^k$ . If the context introduces no confusion then we shall omit the index  $k$ .

### Configuration Space and Poses.

We let  $\mathcal{C}$  denote the *configuration space* of the robot  $A$  in the work space  $\mathcal{W}$  which is scattered with the obstacles  $\{O_k\}$ . The *free* and *forbidden* part of  $\mathcal{C}$  are denoted by  $\mathcal{C}_{\text{free}}$  and  $\mathcal{C}_{\text{forb}}$  respectively. An element  $\mathbf{q} \in \mathcal{C}$  is called a *configuration point*, or *configuration* for short. Given a configuration  $\mathbf{q} \in \mathcal{C}$  we denote by  $A(\mathbf{q})$  the portion of the work space which is covered by  $A$  when it assumes the configuration  $\mathbf{q}$  and it is called either *placement*, or *pose*, or simply *configuration*, when there is no risk of confusion. Similarly  $R_0(\mathbf{q})$ ,  $a_i(\mathbf{q})$  and  $E_i^A(\mathbf{q})$  denote respectively the position, in the work space, of the reference point,  $i$ -th vertex or  $i$ -th edge of the robot. In particular, for a configuration point  $\mathbf{q} = (\vec{r}, \theta)$  with translation component  $\vec{r}$  and rotation component  $\theta$ , we have

$$R_0(\mathbf{q}) = \vec{r} + R_0,$$

$$E_i^A(\mathbf{q}) = \overline{a_{i+1}(\mathbf{q})a_i(\mathbf{q})}.$$

In order to express  $a_i(\mathbf{q})$  for an arbitrary configuration  $\mathbf{q}$ , we have to choose a model of the configuration space. To that end, we consider two possible models

$$\mathcal{C}^{\text{geom}} = \{(x, y, \theta) \mid (x, y) \in \mathbb{R}^2, \theta \in [0, 2\pi)\} \quad (1)$$

$$\mathcal{C}^{\text{rat}} = \{(x, y, \tau) \mid (x, y) \in \mathbb{R}^2, \tau \in \mathbb{R} \cup \infty\} \quad (2)$$

which we call the *geometrical* and *rational* models respectively. Note that

$$\mathcal{C}^{\text{geom}} = \mathbb{R}^2 \times S^1 \text{ and } \mathcal{C}^{\text{rat}} = \mathbb{R}^2 \times \mathbb{RP}^1.$$

These models are related by  $\tau = \tan \frac{\theta}{2}$ . In particular, for  $\mathbf{q} = (\vec{r}, \theta) \in \mathcal{C}^{\text{geom}}$  we have

$$a_i(\mathbf{q}) = \vec{r} + R^\theta a_i \quad (3)$$

with  $R^\theta$  denoting the standard *rotation matrix* and where  $a_i$  is the  $i$ -th vertex of  $A$  in the rest position.

Similarly, for  $\mathbf{q}' = (\vec{r}, \tau) \in \mathcal{C}^{\text{rat}}$  we have

$$a_i(\mathbf{q}') = \vec{r} + M^\tau a_i$$

with the so-called *rational rotation matrix*

$$M^\tau = \frac{1}{1+\tau^2} \begin{pmatrix} 1-\tau^2 & -2\tau \\ 2\tau & 1-\tau^2 \end{pmatrix}.$$

Note that since  $\lim_{\tau \rightarrow \infty} M^\tau = \lim_{\tau \rightarrow -\infty} M^\tau = R^\pi$  we can safely set  $M^\infty = R^\pi$ .

When using the rational model of the configuration space and taking rational coordinates for the translation vector  $\vec{r}$  and letting  $\tau \in \mathbb{Q}$ , it is possible to establish *exact* computations of placements. On the other hand, the geometrical representation is of more use when one is trying to visualize elements of the configuration space.

*Remark 1.* The configuration space in our case, namely the one corresponding to a planar robot that is free to rotate and translate (this kind of robot is also called *holonomic*), is homeomorphic to the *special Euclidean group*  $SE(2)$ . Indeed, the following homeomorphisms hold

$$SE(2) \cong \mathbb{R}^2 \times S^1 \cong \mathbb{R}^2 \times \mathbb{RP}^1.$$

See [LaV06, §4.2] for further details.

## 1.2 Contacts and the Boundary of the Forbidden Space

We say that  $A(\mathbf{q})$  *touches* or is *in contact* with an obstacle  $O$  for a configuration  $\mathbf{q}$  if

$$\partial A(\mathbf{q}) \cap \partial O \neq \emptyset \wedge \text{int}(A(\mathbf{q})) \cap \text{int}O = \emptyset.$$

If only

$$\partial A(\mathbf{q}) \cap \partial O \neq \emptyset,$$

then we say that  $A(\mathbf{q})$  *pseudo touches* or is *in pseudo contact* with the obstacle  $O$ . For a configuration  $\mathbf{q}$ , such that  $A$  pseudo touches *or* just touches an obstacle  $O$ , one or more of the following *contact types* can hold:

Name	Notation	Definition
Vertex-Edge	$(v_i-e_j)$	$a_i(\mathbf{q}) \cap \text{int}E_j^O \neq \emptyset$
Edge-Vertex	$(e_i-v_j)$	$\text{int}E_i^A(\mathbf{q}) \cap b_j \neq \emptyset$
Vertex-Vertex	$(v_i-v_j)$	$a_i(\mathbf{q}) = b_j$
Edge-Edge	$(e_i-e_j)$	$ \text{int}E_i^A(\mathbf{q}) \cap \text{int}E_j^O  > 1$

Note that the contact type alone does not imply whether the interiors of the robot and of the obstacle intersect or not. Note, in addition, that a robot can maintain various pseudo contacts and contacts with the same obstacle simultaneously (cf.  $O_3$  in Figure 2). In the presence of more than one obstacle in  $\mathcal{W}$  the robot can maintain multiple contacts as well. The following definitions refer to portions of  $\mathcal{C}_{\text{forb}}$  which maintain a fixed contact type.

**Definition 1** (Contact Surface). *The set of all configuration points that correspond to a pseudo contact between a fixed vertex (or an edge) of the robot and a fixed vertex (or an edge) of an obstacle is called a contact surface.*

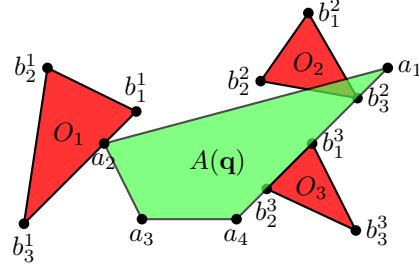


Figure 2: In this example, the configuration  $\mathbf{q}$  corresponds to a  $(v_2-e_3)$  contact with  $O_1$  and a  $(e_4-v_3)$  pseudo contact with  $O_2$ . Finally,  $A(\mathbf{q})$  maintains two  $(v-v)$  contacts and one  $(e-e)$  contact with  $O_3$  simultaneously. This means that  $\mathbf{q}$  belongs to four different contact patches and to one contact surface.

Note that a contact surface is a subset of  $\mathcal{C}_{\text{forb}}$ , since a configuration which realizes a pseudo contact may realize at the same time an intersection of the interiors of the robot and an obstacle. The following definition focuses on the configurations which realize *contacts*.

**Definition 2** (Contact Patch). *The set of all configuration points that correspond to a contact between a fixed vertex (or an edge) of the robot and a fixed vertex (or an edge) of an obstacle is called a contact patch.*

Every contact between a robot and an obstacle is also a pseudo contact, thus each contact patch is a subset of a contact surface. Furthermore, it is a subset of  $\partial \mathcal{C}_{\text{forb}}$ . In addition, the union of all contact patches is the boundary of the forbidden space. Finally, a contact surface which maintains either a  $(v-e)$  or  $(e-v)$  contact type is of dimension two, whereas a contact surface which maintains either a  $(v-v)$  or  $(e-e)$  contact type is of dimension one. This means that the boundary  $\partial \mathcal{C}_{\text{forb}}$  is a union of contact patches of dimension two which are “glued” together with contact patches of dimension one. Figure 2 illustrates the notions discussed in this section.

In this paper we will formulate an explicit parameterization of the contact surfaces depending on the properties of the robot and the obstacles. Furthermore, we will find a subset of the parameter domain, of each contact surface, which corresponds to the respective contact patch. Thus, we will be able to parameterize the whole boundary of the forbidden space.

## 2 ROTATING THE ROBOT

Since the robot  $A$  that we consider is holonomic, every point  $P \in \mathcal{W}$  can be a center of rotation of the robot. In particular, given a configuration point  $\mathbf{q} \in \mathcal{C}$ , the robot can rotate about every boundary point  $\partial A(\mathbf{q}) \in \mathcal{W}$ . This kind of motion is the corner stone of the parameterization that we formalize in this paper.

Let us set a point  $P \in \mathcal{W}$  and mark a point  $a \in \partial A(\mathbf{0})$  on the boundary of the robot. According to the notations that we use,  $a(\mathbf{0}), a(\mathbf{q})$  denote the position of the

marked point when the robot is in either the rest position or in some pose corresponding to a configuration  $\mathbf{q}$ . We will parameterize the set of configuration points in which the marked point  $a$  is fixed to the point  $P$ . More precisely, we want to parameterize the following set

$$P_a = \{\mathbf{q} \in \mathcal{C} : a(\mathbf{q}) = P\}.$$

**Lemma 1.** *Given a point  $P \in \mathcal{W}$  and a point  $a \in \partial A$  the set  $P_a$  is parameterized by*

$$\mathbf{q}_a(\phi) = \begin{pmatrix} \vec{r}_a(\phi) \\ \theta_a(\phi) \end{pmatrix} = \begin{pmatrix} P - R^\phi a \\ \phi \end{pmatrix} \quad (4)$$

for  $\phi \in [0, 2\pi)$ . That is,  $\mathbf{q} \in P_a$  if and only if  $\mathbf{q} = \mathbf{q}_a(\phi)$  for some  $\phi \in [0, 2\pi)$ .

*Proof.* Since  $a \in E_i^A$  for some index  $i$ , we can write  $a = (1-t)a_i + ta_{i+1}$  for some  $t \in [0, 1)$ . First we show that if  $\mathbf{q} = \mathbf{q}_a(\phi)$  for some  $\phi$  then  $\mathbf{q} \in P_a$ , that is  $a(\mathbf{q}) = P$ . For every  $\phi$ , using Equation (3), we have

$$\left. \begin{aligned} a_i(\mathbf{q}_a(\phi)) &= P - R^\phi a + R^\phi a_i \\ a_{i+1}(\mathbf{q}_a(\phi)) &= P - R^\phi a + R^\phi a_{i+1} \end{aligned} \right\}.$$

It is easy to show that

$$a(\mathbf{q}_a(\phi)) = (1-t)a_i(\mathbf{q}_a(\phi)) + ta_{i+1}(\mathbf{q}_a(\phi)) = P.$$

That is, for every  $\phi$ , the point  $a(\mathbf{q}_a(\phi))$  is fixed to  $P$ .

Conversely, given  $\mathbf{q} = (\vec{r}, \theta) \in P_a$  we have

$$P = a(\mathbf{q}) = \vec{r} + R^\theta a.$$

Thus,  $\vec{r} = P - R^\theta a$ . Finally, for  $\phi = \theta$  we have that  $\mathbf{q} = \mathbf{q}_a(\phi)$ .  $\square$

We observe that for  $\phi = 0$  the parameterization given in Equation (4) is merely a translation. That is, the local frame which is assigned to the robot for  $\mathbf{q}_a(0)$  is aligned with the global frame of the work space. At this point it is important to point out that  $\mathbf{q}_a(\phi)$  is meaningful only when interpreted as a point in  $\mathcal{C}^{\text{geom}}$ . Finally, Equation (4) is a parameterization of a *helix* in the configuration space. We conclude this section with the following remark.

**Remark 2** (Covering  $\mathcal{C}$ -Space with Helices). Instead of taking  $a \in \partial A$ , we can generalize the idea and consider an arbitrary linear combination of the vertices of the robot,  $a = \sum_{i=1}^n \lambda_i a_i$ , and some point  $P \in \mathcal{W}$ . The set of configurations which correspond to a rotation of the robot such that  $a$  is fixed to  $P$  is again a helix. As a matter of fact, every configuration point  $\mathbf{q} \in \mathcal{C}$  is contained in infinitely many helices of this form.

### 3 PARAMETERIZING CONTACT SURFACES

In this section we consider the robot  $A$  and *one convex* obstacle  $O$ . Later, an arbitrary obstacle can be decomposed into convex subsets and each sub-obstacle can be treated in a similar way. Given a contact type of  $A$  and  $O$  we will derive an explicit parameterization of the corresponding contact surface and patch.

#### 3.1 Vertex-Edge Contact

A  $(v_i-e_j)$  vertex-edge pseudo contact occurs when a vertex  $a_i$  of the robot lies in the interior of an edge  $E_j^O$  of the obstacle (see  $O_1$  and  $A(\mathbf{q})$  in Figure 2 for an example). In this section, we utilize the parameterization obtained in Section 2, and provide an explicit parameterization of the contact surface and the contact patch in the configuration space corresponding to the prescribed  $(v_i-e_j)$  pseudo contact and contact respectively.

Let  $P(t) = (1-t)b_j + tb_{j+1}$  be an arbitrary point in the interior of  $E_j^O$ . The configurations that correspond to the  $(v_i-e_j)$  contact can be derived from Equation (4) by replacing  $P$  with  $P(t)$  and  $a$  with  $a_i$  and is given by

$$\begin{aligned} S(t, \phi) &= \begin{pmatrix} P(t) - R^\phi a_i \\ \phi \end{pmatrix} \\ &= c(\phi) + t\vec{r}(\phi), \end{aligned} \quad (5)$$

for  $t \in (0, 1)$  and  $\phi \in [0, 2\pi)$ . As  $\phi$  varies in the interval  $[0, 2\pi)$ , the configuration points on  $S$  represent both contacts and pseudo contacts. Clearly, this surface is a *ruled surface* with directrix  $c(\phi)$  and  $\vec{r}(\phi) \neq 0$  as the vector field. Note that  $\frac{d}{d\phi}\vec{r}(\phi) = 0$  which implies that  $S$  is a *cylindrical ruled surface* and thus *developable* [dC76]. It is easy to verify that  $S(t, \phi)$  is a collection of congruent helices. Note that for  $t \in \{0, 1\}$  the parameterization reduces to two helices which correspond to the two pseudo contacts  $(v_i-v_j)$  and  $(v_i-v_{j+1})$  respectively (cf. Section 3.3). In Figure 7, such a surface is illustrated with helical arcs in black and rulings in yellow.

**Remark 3.** If we fix a vertex  $a_i$  of the robot and generate all possible vertex-edge contact surfaces with all edges of the obstacle, then *helices* contained in each of these contact surfaces are congruent copies of each other. Note that if the obstacles are regular polygons then for a fixed vertex of the robot the contact surfaces are just congruent copies of each other. If, in addition, the robot is a regular polygon then *all the vertex-edge contact surfaces* are congruent copies of each other.

Our next goal is to find a sub-domain  $\Phi \subset [0, 2\pi)$  such that  $S(t, \phi)|_{\phi \in \Phi}$  will be the *contact patch* which is contained in  $S$ . In Section 3.1.1 we analyze the domain  $[0, 2\pi)$  of  $\phi$  and find this sub-domain  $\Phi$ .

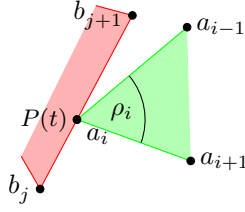


Figure 3: Geometrical components of the  $(v_i-e_j)$  contact.

### 3.1.1 Vertex-Edge Angle Range Analysis

By now, given a vertex  $a_i$  of the robot and an edge  $E_j^O$  of an obstacle  $O$ , we have an explicit parameterization of the contact surface  $S$  in the configuration space which corresponds to the  $(v_i-e_j)$  pseudo contact. Our goal is to find a contact patch  $S' \subset S \in \mathcal{C}$ , such that for all  $\mathbf{q} \in S'$  we will have that  $a_i(\mathbf{q})$  touches the edge of the obstacle.

Since we assume that both the robot and the obstacle are convex, the sub-domain  $\Phi$  is independent of  $t$ , namely independent of the point of contact along  $E_j^O$ . This can be seen in Figures 2 and 3. For some fixed  $t_0 \in (0, 1)$  let  $\mathbf{q}_i(\phi) = S(t_0, \phi)$  be a helix in  $S$  which corresponds to the pseudo contact between  $a_i$  and  $P = P(t_0)$ . The sub-domain  $\Phi$  can be determined by finding two values:

- $\phi_{\min}$ : The minimal angle for which  $a_{i+1}(\mathbf{q}_i(\phi_{\min}))$  lies on the line containing the edge  $E_j^O$  and simultaneously  $a_{i-1}(\mathbf{q}_i(\phi_{\min}))$  lies to the right of this edge.
- $\phi_{\text{range}}$ : The range of rotation that maintains the contact of  $a_i$  with  $P$ . In practice this means that we want  $a_{i-1}(\mathbf{q}_i(\phi_{\min} + \phi_{\text{range}}))$  to lie on the line containing  $E_j^O$  such that  $a_{i+1}(\mathbf{q}_i(\phi_{\min} + \phi_{\text{range}}))$  will lie to its right. See Figure 3 for an illustration.

If we let  $\phi_{\max} = \phi_{\min} + \phi_{\text{range}}$ , then

$$\Phi = \begin{cases} [\phi_{\min}, \phi_{\max}] & \text{if } \phi_{\max} < 2\pi \\ [\phi_{\min}, 2\pi] \cup [0, \phi_{\max} - 2\pi] & \text{if } \phi_{\max} \geq 2\pi \end{cases} \quad (6)$$

is the sub-domain we want to find.

We now compute the values of  $\phi_{\min}$  and  $\phi_{\text{range}}$ . The latter is straightforward to find, and depends on the interior angle at the vertex  $a_i$  of  $A$ , namely

$$\phi_{\text{range}} = \pi - \rho_i.$$

#### Computing $\phi_{\min}$ .

We want to find  $\phi$  such that for  $\mathbf{q}_i(\phi) \in \mathcal{C}$  the following will hold:

$$P - \|E_i^A\| \frac{E_j^O}{\|E_j^O\|} = a_{i+1}(\mathbf{q}_i(\phi)),$$

$x_0$	$y_0$	$\phi_{\min} \in$
$\geq 0$	$\geq 0$	$[0, \frac{\pi}{2}]$
$< 0$	$\geq 0$	$[\frac{\pi}{2}, \pi]$
$< 0$	$< 0$	$[\pi, \frac{3\pi}{2}]$
$\geq 0$	$< 0$	$[\frac{3\pi}{2}, 2\pi]$

Table 1: Interval of  $\phi_{\min}$  depending on signs of  $x_0$  and  $y_0$  for the vertex-edge and edge-vertex contact types.

where  $E_j^O$  is consider as the vector from  $b_j$  to  $b_{j+1}$  and  $\|\cdot\|$  denotes the length of an edge. Solving this equation for  $\phi$  is equivalent to solving

$$-\|E_i^A\| \frac{E_j^O}{\|E_j^O\|} = M \cdot (x, y)^T,$$

where  $x = \cos \phi$ ,  $y = \sin \phi$  and

$$M = \left( E_i^A, R^{\frac{\pi}{2}} \cdot E_i^A \right)^T. \quad (7)$$

Since  $\det M = \|E_i^A\|^2 \neq 0$ , this system has a unique solution, denoted by  $(x_0, y_0)^T$ . We define  $\{\phi_i\}_{i=1}^4$  as follows

$$\begin{aligned} \{\phi_1, \phi_2\} &= \arccos(x_0) \cap [0, 2\pi) \\ \{\phi_3, \phi_4\} &= \arcsin(y_0) \cap [0, 2\pi) \end{aligned}$$

Note that since  $(x_0, y_0)$  is a unit vector, we have that  $\{\phi_1, \phi_2\} \cap \{\phi_3, \phi_4\}$  contains exactly one element. As  $\phi_{\min}$  should lie in  $[0, 2\pi)$ , it satisfies

$$\phi_{\min} = \{\phi_1, \phi_2\} \cap \{\phi_3, \phi_4\}.$$

For any combination of signs of  $x_0$  and  $y_0$  Table 1 suggests in which interval  $\phi_{\min}$  is, and using the definition of the  $\phi_i$ 's it can be easily found.

Finally, in Figure 9 we plot an example of all possible contact patches which correspond to a triangular robot and obstacle. The red patches are the vertex-edge contact patches. Let us conclude this section with one remark.

*Remark 4* (On the exactness of computations). The steps that we described so far, in general, cannot yield the exact value of  $\phi_{\min}$  since one has to compute the inverse functions of both sine and cosine. Furthermore, the matrix  $M$  in Equation (7) involves the trigonometric functions as well, and thus cannot be represented in an exact manner. In turn, this means that  $x_0$  and  $y_0$  above cannot be computed exactly in the first place. If the vertices of the robot are assumed to lie on a circle of some fixed radius, then it is possible to find  $\Phi$  without trigonometric functions; further details can be found in [AGR13].

## 3.2 Edge-Vertex Contact

Recall that Equation (4) parameterizes a rotation of the robot about a point  $P$  such that a boundary point  $a \in \partial A$  is fixed to  $P$ . For any  $t \in (0, 1)$  we denote

$$a_{i,t} = (1-t)a_i + ta_{i+1}$$

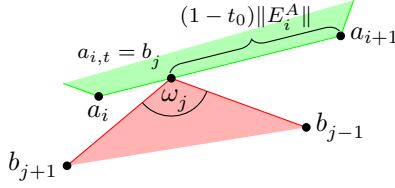


Figure 4: Geometrical components of the  $(e-v_j)$  contact.

to be the point on the edge  $E_i^A$  of the robot which will be in pseudo contact with the vertex  $b_j$  of the obstacle. Next, in Equation (4), we replace  $P$  with  $b_j$  and  $a$  with  $a_{i,t}$  and obtain

$$\begin{aligned} S(t, \phi) &= \begin{pmatrix} b_j - R^\phi a_{i,t} \\ \phi \end{pmatrix} \\ &= c(\phi) + t\vec{r}(\phi) \end{aligned} \quad (8)$$

for  $t \in (0, 1)$  and  $\phi \in [0, 2\pi)$ .  $S(t, \phi)$  is the contact surface corresponding to the  $(e-v_j)$  pseudo contact. Again, like the parameterization in Equation (5) we obtain a ruled surface, obviously with different directrix and vector field. This surface is swept by a horizontal line segment which translates and rotates in  $\mathcal{C}$ . In Figure 8 an example of a typical edge-vertex contact surface is plotted with helices in black and rulings in yellow.

**Remark 5.** In contrast to the case of  $(v-e)$  contact surfaces, here in the edge-vertex case, each contact surface is a collection of helices which are *not* congruent since their radii depend on the varying  $a_{i,t}$ . This suggests that  $(e-v)$  contact surfaces are *not* developable. We establish this fact and study the geometry of these surfaces in Section 4.

In order to find the contact *patch* which is contained in  $S$ , as before, we have to find a sub-domain  $\Phi \subset [0, 2\pi)$  for which  $S(t, \phi)_{|\phi \in \Phi}$  is a collection of configuration points which correspond to contacts and *not* to pseudo contacts. Again, as can be seen in Figure 4, the sub-domain  $\Phi$  does not depend on  $t$ .

### 3.2.1 Edge-Vertex Angle Range Analysis

In this section, similarly to the procedure discussed in Section 3.1.1, we find  $\phi_{\min}$ ,  $\phi_{\text{range}}$  and  $\phi_{\max}$ , where  $\phi_{\max} = \phi_{\min} + \phi_{\text{range}}$ , such that for  $\Phi$  as defined in Equation (6) the sub-surface  $S(t, \phi)_{|\phi \in \Phi}$  is a *contact patch*. As  $\phi_{\min}$  and  $\phi_{\text{range}}$  depend only on the indices  $i$  and  $j$  we shall fix some  $t_0 \in (0, 1)$  and let  $\mathbf{q}_i(\phi) = S(t_0, \phi)$ . In this case,  $a_{i+1}(\mathbf{q}_i(\phi_{\min}))$  has to lie on the line containing  $E_{j-1}^O$ , such that the interiors of the robot and the obstacle do not intersect. Similarly,  $a_i(\mathbf{q}_i(\phi_{\max}))$  has to lie on the line segment containing  $E_j^O$  (cf. Figure 4). Clearly, we have that

$$\phi_{\text{range}} = \pi - \omega_j.$$

It is left to find the value of  $\phi_{\min}$ .

### Computing $\phi_{\min}$ .

In the case of  $(e-v)$  contact, the minimal angle of rotation  $\phi_{\min}$  is the one for which the following will hold

$$a_{i+1}(\mathbf{q}_i(\phi)) - b_j \parallel b_j - b_{j-1}. \quad (9)$$

The condition in (9) together with the restriction that  $a_{i,t_0}$  has to coincide with  $b_j$  can be formulated as follows:

$$b_j - (1 - t_0) \|E_i^A\| \frac{E_{j-1}^O}{\|E_{j-1}^O\|} = a_{i+1}(\mathbf{q}_i(\phi)),$$

with  $\phi$  as the unknown. Computations similar to those we described in Section 3.1.1 can be applied in this case as well. This last equation can be rewritten as follows

$$(t_0 - 1) \|E_i^A\| \frac{E_{j-1}^O}{\|E_{j-1}^O\|} = M \cdot (x, y)^T$$

with  $x = \cos \phi$ ,  $y = \sin \phi$  and

$$M = \left( a_{i+1} - a_{i,t_0}, R^{\frac{\pi}{2}} \cdot (a_{i+1} - a_{i,t_0}) \right)^T.$$

As before, this system has a unique solution denoted by  $(x_0, y_0)^T$ . Based on the cases given in Table 1 we can find a unique solution  $\phi_{\min}$ .

We conclude the section by referring to the green patches in Figure 9 which correspond to  $(e-v)$  contacts.

## 3.3 Vertex-Vertex and Edge-Edge Contacts

In the previous sections we found explicit parameterizations of the contact surfaces and contact patches of dimension two, namely those that correspond to either vertex-edge or edge-vertex pseudo contacts and contacts. In order to complete the picture we have to consider the configurations that correspond to vertex-vertex and edge-edge pseudo contacts and contacts. To that end, we recall that the boundaries of the two dimensional contact surfaces that we have derived are exactly the one dimensional contact surfaces.

In the example plotted in Figure 9 blue helical arcs correspond to vertex-vertex contacts and the yellow (straight) lines correspond to edge-edge contacts.

### 3.3.1 Vertex-Vertex Contact

For two indices  $i$  and  $j$ , the contact surface which corresponds to the  $(v_i-v_j)$  pseudo contact is parameterized by

$$C(\phi) = S(0, \phi),$$

for  $\phi \in [0, 2\pi)$  and  $S(\cdot, \cdot)$  as given in Equation (5). In order to find the sub-domain  $\Phi$ , which corresponds to the  $(v_i-v_j)$  contacts alone, we have to compute  $\phi_{\min}$



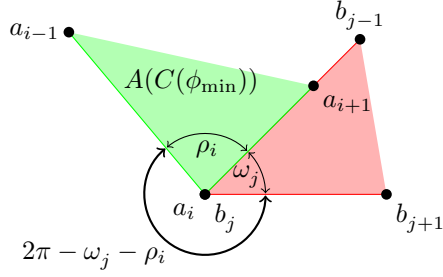


Figure 5: Illustration of a vertex-vertex contact with  $C(\cdot)$  and  $\phi_{\min}$  as defined in Section 3.3.1.

that corresponds to the  $(v_i-e_{j-1})$  contact. For the pose which corresponds to  $C(\phi_{\min})$  we have that  $a_i$  coincides with  $b_j$ ,  $a_{i+1}$  lies on the line containing  $E_{j-1}^O$  and  $a_{i-1}$  lies to the right of this edge. See Figure 5 for an illustration. Letting  $\phi$  vary in the sub-domain  $\Phi = [\phi_{\min}, \phi_{\min} + 2\pi - \omega_j - \rho_i]$  yields the helical sub-arc of  $C(\phi)$  which maintain a  $(v_i-v_j)$  contact. As before, if  $\phi_{\min} + 2\pi - \omega_j - \rho_i \geq 2\pi$  then the sub-domain  $\Phi$  is the union  $[\phi_{\min}, 2\pi) \cup [0, \phi_{\min} - \omega_j - \rho_i]$ .

### 3.3.2 Edge-Edge Contact

Parameterizing the configurations, which correspond to an  $(e_i-e_j)$  contact, can be again obtained using the parameterization of a corresponding  $(v_i-e_j)$  contact. Let us set

$$C(t) = S(t, \phi_{\min}),$$

where  $S(\cdot, \cdot)$  is given in Equation (5) and  $\phi_{\min}$  is the one defined in Section 3.1.1. In this case  $C(0)$  corresponds to a  $(v_i-v_j)$  contact and  $C(1)$  corresponds to a  $(v_i-v_{j+1})$  contact. In both cases  $a_{i+1}$  lies on the line containing  $E_j^O$ . See Figure 6 for reference where it is shown that for  $t \in [0, 1)$  we do not obtain the whole  $(e_i-e_j)$  contact. In order to complete the case, we have to let

$$t \in \left[0, 1 + \frac{\|E_i^A\|}{\|E_j^O\|}\right).$$

## 3.4 Summary of the Parameterization

In this section we derived, based on the fundamental motion described in Section 2, the parameterization of all the elements of the boundary of the forbidden space which correspond to a single obstacle. If the work space contains more obstacles, each one of them contributes another pillar-like element similar to the one depicted in Figure 9. Given an obstacle  $O$ , the portion of  $\mathcal{C}$  which is bounded “inside” the corresponding pillar-like object is the forbidden space related to  $O$ .

## 4 DIFFERENTIAL GEOMETRY OF CONTACT SURFACES

Using the parameterization that we developed, we study in this section the geometrical properties of the contact

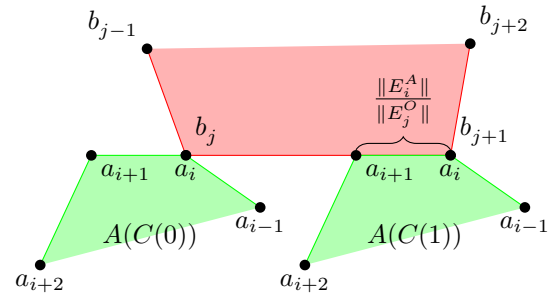


Figure 6: Setting of an edge-edge contact where  $C(\cdot)$  is given in Section 3.3.2.

surfaces. As we already pointed out, the contact surfaces which correspond to vertex-edge contacts are developable and thus rather simple. We, therefore, focus in this section on the case of edge-vertex contact surfaces.

For the sake of simplicity, in Equation (8), we assume that  $b_j$  is at the origin. Thus we consider the contact surface  $S(t, \phi) = c(\phi) + t\vec{r}(\phi)$  with

$$c(\phi) = \begin{pmatrix} -R^\phi a_i \\ \phi \end{pmatrix}, \quad \vec{r}(\phi) = \begin{pmatrix} R^\phi (a_i - a_{i+1}) \\ 0 \end{pmatrix}.$$

Note that  $\vec{r}(\phi), \frac{d}{d\phi}\vec{r}(\phi) \neq 0$ ; this means that the  $(e_i-v_j)$  contact surface is a *non-cylindrical ruled surface* [dC76].

We start our study of the geometrical properties of the contact surface by computing its first (denoted  $E, F, G$ ) and second (denoted  $e, f, g$ ) fundamental forms:

$$\begin{aligned} E &= \|a_i - a_{i+1}\|^2 & e &= 0 \\ F &= \det(a_i, a_{i+1}) & f &= -\frac{\|a_i - a_{i+1}\|^2}{v} \\ G &= 1 + \|a_{i,t}\|^2 & g &= -\frac{\det(a_i, a_{i+1})}{v} \end{aligned}$$

where

$$v = v(t) = \sqrt{EG - F^2}.$$

It is easy to verify that  $v \neq 0$  for all  $t \in \mathbb{R}$ , and thus, all expressions are well defined. Using standard formulas we can find the curvatures of the surface.

**Lemma 2.** *The Gaussian curvature  $K(t, \phi)$  and the mean curvature  $H(t, \phi)$  of an  $(e-v)$  contact surface are given by:*

$$K(t, \phi) = -\frac{E^2}{v^4}, \quad H(t, \phi) = \frac{EF}{2v^3}.$$

Note that both the Gaussian and the mean curvature do not depend on  $\phi$  but *only* on  $t$  as  $v$  depends on  $t$ . This comes as no surprise, as the surface is ruled. In other words, these curvatures depend on the point of pseudo contact along  $E_i^A$ . Lemma 2 also proves that the  $(e-v)$

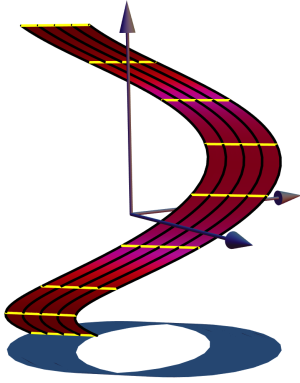
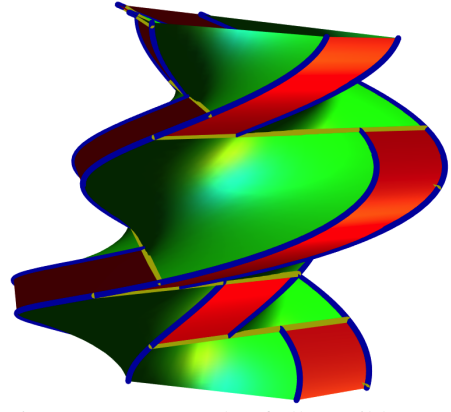


Figure 7: (v-e) contact surface.



Figure 8: (e-v) contact surface.

Figure 9: An example of all possible contact patches for a given robot and one obstacle in  $\mathcal{C}$ .

contact surface is not developable since  $K(t) < 0$ . The next lemma establishes the cases when the edge-vertex contact surface is a *minimal surface*, and in turn also a *helicoid*.

**Lemma 3.** *If the line through  $E_i^A(\mathbf{0})$  contains the reference point  $R_0(\mathbf{0})$ , then the corresponding  $(e_i-v_j)$  contact surface is a minimal surface.*

*Proof.* Recall that we assume that  $R_0(\mathbf{0})$  is at the origin. Thus, if the line through  $E_i^A(\mathbf{0})$  contains  $R_0(\mathbf{0})$ , then  $a_i$  and  $a_{i+1}$ , as vectors, are linearly dependent. Therefore  $F = 0$ , and in turn  $H(t) = 0$ .  $\square$

It is easy to verify that  $\frac{dK}{dt}(t_*) = \frac{dH}{dt}(t_*) = 0$  for

$$t_* = \frac{\langle a_i, a_i - a_{i+1} \rangle}{E}.$$

The *striction curve* [PW01, dC76] can now be found. Indeed, since  $\lim_{t \rightarrow \pm\infty} K(t) = 0$ ,  $K(t) < 0$  and  $K'(t)$  vanishes only for  $t_*$ , it turns out that the Gaussian curvature attains its global extremum along the curve  $S(t_*, \phi)$ , which is therefore the striction curve. Note however, that  $t_*$  may not lie in the interval  $[0, 1]$ , that is, the striction curve of the (e-v) contact surface is not necessarily contained in it.

Next, we will compute the *normal curvature* of the (e-v) contact surface. Let  $\varepsilon(\xi) \in T_p S$  be a unit tangent vector parameterized by a direction  $\xi$ . In turn, the *normal curvature* in direction  $\xi$ , denoted by  $\kappa_N(\xi)$ , is given by [Gra93]

$$\kappa_N(\xi) = \frac{E \sin \xi (F \sin \xi - 2v \cos \xi)}{v^3}. \quad (10)$$

Recall that  $v$  does not vanish for all  $t \in \mathbb{R}$ , and therefore Equation (10) is well defined. Note that the normal curvature depends only on the direction in the tangent plane, given by  $\xi$ , and the position on the ruling given by  $t$ .

Next, we want to find the *asymptotic directions* and the *principal curvature directions* of the contact surface. In other words we want to find the values of  $\xi$  (parameterized by  $t$ ) for which the normal curvature either vanishes or attains an extremum. Furthermore, we will find the extremal values of the normal curvature, that is, expressions of the *principal curvatures*.

As the rulings on the surface are straight lines, the normal curvature in their directions should vanish, and indeed for  $\xi \in \{0, \pi\}$  the normal curvature vanishes. The other direction where it vanishes corresponds to

$$\xi = \arctan \frac{2v}{F}.$$

It is easy to verify that  $\frac{d}{d\xi} \kappa_N(\xi)$  vanishes for

$$\xi_1 = \frac{1}{2} \arctan \frac{2v}{F}.$$

and the normal curvature attains an extremum in this direction. Since the principal curvature directions are orthogonal, the normal curvature attains its other extremum for  $\xi_2 = \xi_1 + \frac{\pi}{2}$ . Finally, the principal curvatures are given by

$$\kappa_1 = \kappa_N(\xi_1) \quad , \quad \kappa_2 = \kappa_N(\xi_2).$$

One can verify that  $K = \kappa_1 \cdot \kappa_2$ . In Figure 10 a small sub-surface of an (e-v) contact surface is plotted together with the asymptotic and principal curvature directions.

*Remark 6.* The principal curvatures could have been computed directly using formulas which uses the fundamental forms. We took a longer path as we wanted to obtain expressions for the principal curvature directions as well.

## 5 CONCLUSION

### Geometrical vs. Rational Models of $\mathcal{C}$ .

In this paper we considered  $\mathcal{C}^{\text{geom}}$ , see Equation (1), as the model of the configuration space. Using this model



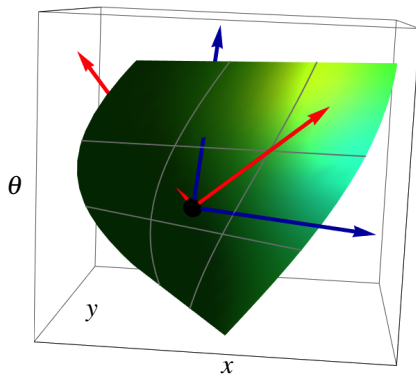


Figure 10: A sub-surface of an (e-v) contact surface. In blue and red the asymptotic and principal curvatures directions at some arbitrary point on the surface are respectively plotted.

it is easy to visualize elements in  $\mathcal{C}$  (cf. Figures 7 to 10). This, however, comes with a price. The parameterization of the contact surfaces or patches themselves, given in Equations (5) and (8), involves the trigonometric functions and thus it could not be computed in an exact manner. Furthermore, the computations of the sub-domains that correspond to the contact *patches* involve again trigonometric functions and their inverses, thus, once more, the final result cannot have an exact representation. This inexactness can, for example, have implications when one is using the parameterization to compute intersections between contact surfaces or patches. However, it is easy to replace  $\mathcal{C}^{\text{geom}}$  with  $\mathcal{C}^{\text{rat}}$ , see Equation (2); this change can yield *rational* and *exact* representation of the contact surfaces. The trade-off in this case is that visualizing the contact surfaces in  $\mathcal{C}^{\text{rat}}$  is not intuitive.

### Applications.

Using the parameterization described in this paper we produced a short video which visualizes the configuration space of a convex polygonal robot which moves amid convex polygonal obstacles in the plane. This video is available online [AR12].

### Future Work.

The computations of the contact *patches* heavily rely on the assumption that the robot  $A$  is convex. Eliminating this restriction can be of interest. Other interesting extensions of the parameterization would be to consider a robot with a boundary which consists of non-linear edges; for example circular arcs (cf. [MST13]).

Once the contact surfaces and patches can be explicitly parameterized, it is natural to consider their discretizations. An approximated version of the configuration space can be used to address the general problem of

*motion planning*. Furthermore, given either the smooth or discrete representations of the contact surfaces or patches, it is possible to study their mutual intersections and their intersections with other elements in the configuration space, as done for example in [SHRH11]. This study can be of help in the investigation of the arrangement of the contact surfaces in the configuration space.

## 6 ACKNOWLEDGMENTS

This work is part of the project *Computational Geometric Learning*. The project CG Learning acknowledges the financial support of the Future and Emerging Technologies (FET) program within the Seventh Framework Program for Research of the European Commission, under FET-Open grant number 255827.

## 7 REFERENCES

- [AGR13] Dror Atariah, Sunayana Ghosh, and Günter Rote. On the parameterization and the geometry of the configuration space of a single planar robot. Technical report, Freie Universität Berlin, 2013. In preparation.
- [AR12] Dror Atariah and Günter Rote. Configuration space visualization. In *Proceedings of the 2012 symposium on Computational Geometry*, SoCG '12, pages 415–416, New York, NY, USA, 2012. ACM. <http://youtu.be/SBFwgR4K1Gk>.
- [BA88] Jean-Daniel Boissonnat and Francis Avnaim. Polygon placement under translation and rotation. Rapport de recherche RR-0889, INRIA, 1988.
- [CLH<sup>+</sup>05] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series)*. MIT Press, 2005.
- [dC76] Manfredo P. do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1976. Translated from the Portuguese.
- [Gra93] Alfred Gray. *Modern Differential Geometry of Curves and Surfaces*. CRC Press, 1993.
- [HA92] Yong K. Hwang and Narendra Ahuja. Gross motion planning – A survey. *ACM Comput. Surv.*, 24(3):219–291, September 1992.
- [IKP73] M.B. Ignat'ev, F.M. Kulakov, and A.M. Pokrovskii. *Robot-manipulator control algorithms*. Joint Publications Research Service, 1973.

- [Lat93] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 3rd edition, 1993.
- [Lau98] Jean-Paul Laumond, editor. *Robot Motion Planning and Control*, volume 229 of *Lectures Notes in Control and Information Sciences*. Springer, 1998.
- [LaV06] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at <http://planning.cs.uiuc.edu/>.
- [LPW79] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, October 1979.
- [MST13] Victor Milenkovic, Elisha Sacks, and Steven Trac. Robust complete path planning in the plane. In Emilio Frazzoli, Tomas Lozano-Perez, Nicholas Roy, and Daniela Rus, editors, *Algorithmic Foundations of Robotics X*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 37–52. Springer Berlin Heidelberg, 2013.
- [PW01] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer-Verlag New York, Inc., 2001.
- [SHRH11] Oren Salzman, Michael Hemmer, Barak Raveh, and Dan Halperin. Motion planning via manifold samples. In *ESA*, pages 493–505, 2011.
- [SS83] J.T. Schwartz and M. Sharir. On the piano movers’ problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.
- [WB00] Kevin D. Wise and Adrian Bowyer. A survey of global configuration-space mapping techniques for a single robot in a static environment. *The International Journal of Robotics Research*, 19(8):762–779, 2000.

# Optimal RANSAC - Towards a Repeatable Algorithm for Finding the Optimal Set

Anders Hast  
Uppsala University,  
Uppsala, Sweden  
anders.hast@it.uu.se

Johan Nysjö  
Uppsala University,  
Uppsala, Sweden  
johan.nysjo@it.uu.se

Andrea Marchetti  
IIT, CNR  
Pisa, Italy  
andrea.marchetti@iit.cnr.it

## ABSTRACT

A novel idea on how to make RANSAC repeatable is presented, which will find the optimal set in nearly every run for certain types of applications. The proposed algorithm can be used for such transformations that can be constructed by more than the minimal points required. We give examples on matching of aerial images using the Direct Linear Transformation, which requires at least four points. Moreover, we give examples on how the algorithm can be used for finding a plane in 3D using three points or more. Due to its random nature, standard RANSAC is not always able to find the optimal set even for moderately contaminated sets and it usually performs badly when the number of inliers is less than 50%. However, our algorithm is capable of finding the optimal set for heavily contaminated sets, even for an inlier ratio under 5%. The proposed algorithm is based on several known methods, which we modify in a unique way and together they produce a result that is quite different from what each method can produce on its own.

## Keywords

RANSAC, Local Optimisation, Repeatable, Optimal Set, Feature Matching, Image Stitching, 3D Planes.

## 1 INTRODUCTION

In aerial image stitching based on feature matching it is necessary to find corresponding points between two or more images. Hence, it must be determined which points are matching, so called inliers, and which points are false matches, so called outliers. RANSAC [FB81] is one of the far most used algorithms for this purpose and many variants have been proposed in literature. The main disadvantage with standard RANSAC is that it is not repeatable [Zul09] since it is based on random sampling, as the name itself suggests: RANdom SAMple Consensus. Therefore, it is difficult using RANSAC while trying to run tests of other parameters involved in the application, as the set of inliers for the same pair of images may vary in each run. For medical applications such as the one described later it is important that the result does not differ if the application is run more than once. Furthermore, standard RANSAC does not try to find the optimal set of inliers, instead it stops when the probability of finding more inliers reaches its predefined threshold. The consequence is that it does not perform well for highly contaminated sets, i.e. when the set of outliers is large, since consensus might not be reached within reason-

able time. Even worse is that the stopping criterion, which is based on a statistical hypothesis might indicate that consensus is not yet reached, while most of the inliers are already found.

Generally the set of inliers is used to construct the geometric transformation or homography [VL01] between the pair of images and because of the randomness in RANSAC it will be a bit different each time. Clearly, if some important inliers are missed, the homography will be different from the best one. Especially if the set is close to degenerate it might not even be useful for image stitching etc as the transformed images will be distorted.

## Contributions and Delimitations

In this paper several known methods (LO-RANSAC [CMK03] [CMO04] [LMC12]) are modified and put together in a unique way giving an algorithm that is repeatable. This means that it will yield the same result in each run, i.e. the optimal set, which are all inliers below a certain threshold defined by the tolerance  $\epsilon$ . When testing different settings of parameters in an application that uses RANSAC it is important that it behaves in a predictable way and will not add unwanted bias to the tests. Furthermore the new algorithm is able to handle sets with a very low inlier ratio, even under 5%, which is important since many implementations of RANSAC do not perform well when the number of inliers is less than 50% [Low04]. The proposed algorithm will generally be faster for such sets but slower for high inlier ratio sets.

The algorithm only works for transformations that can be constructed from more than the minimal points

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

required, such as the 4 point Direct Linear Transformation (DLT) [HZ03].

It will be shown that the algorithm performs well for aerial images but might find a local maxima, i.e. a sub optimal set for photos taken on the ground for stitching panoramas. Moreover, examples of finding a plane in 3D will be given from a medical application, where the transformation is given by the plane equation.

Nonetheless, we have not yet tested the algorithm for finding clusters of points or lines in a point set. As long as there is a way to find a best fit to many points by for instance computing the minimal squared distance or doing Principal Component Analysis (PCA) it should work for such cases, as long as there are not many suboptimal sets present. For Homography transformations between a pair of images we compute the minimal squared distance and for finding the best fitting plane in 3D we use PCA as it turns out to be much faster than computing the minimal squared distance.

## RANSAC and some of its Variants

Some of the many variants of RANSAC used for finding true matches between pairs of images are briefly discussed in this section and we will start by explaining the main idea of the standard RANSAC. As mentioned in the introduction RANSAC is used to determine a set of inliers and first starts by selecting the minimal number of points required to determine the model parameters, i.e. finding the homography [HZ03] [BL07], which is the projective transformation between the images. Then the set is rescored using this transformation, in the way that the number of inliers that falls below a certain predefined tolerance  $\epsilon$ , are counted. This means that when transformed, these points are being close enough to its corresponding match and are hence regarded as true inliers.

If the number of inliers is large enough or more commonly when the probability of finding a better model becomes lower than some threshold, then the algorithm terminates, otherwise it starts all over. Generally,  $N$  iterations are needed in order to find an outlier free set with the probability  $p$  (often set to 99% or more) as:

$$N = \frac{\log(1-p)}{\log(1-\gamma^s)}, \quad (1)$$

where  $\gamma$  is the inlier ratio, i.e. number of inliers divided by number of points in the cluster and  $s$  is the number of samples drawn each time. If  $N$  is larger than the number of iterations of the main loop the algorithm starts all over and samples the set once again. Alternatively one can also run the algorithm several times and then choose the solution giving the largest set of inliers. The main idea however is to generate a hypothesis from random samples (estimating a model) and then verifying it using all the data (scoring). Usu-

ally a final reestimation is performed where the transformation is computed using all inliers [HZ03].

RANSAC generally treats all correspondences equally and draws random samples uniformly from the full set while MLESAC [TZ00] performs non-uniform, i.e. guided sampling of correspondences and PROSAC [CM05] draw samples from progressively larger sets of top-ranked correspondences. GOOD-SAC [MvHK\*06] on the other hand does not use random sampling, but instead an assessment driven selection of good samples. SCRAMSAC [SLK09] tries to reduce the number of outliers using a spatial consistency check in order to find inliers more rapidly. They all aim at reducing those outliers that somehow can be recognized as true mismatches and not as being close to a true match.

A randomized model verification strategy for RANSAC, R-RANSAC [CM08], was proposed for the situation when the contamination of outliers is known. The LO-RANSAC [CMK03] [CMO04] utilizes a local optimization step and when applied to selected models the algorithm has near perfect agreement with the theoretically optimal performance. Another approach [RFP09], known as Cov-RANSAC incorporates the inherent uncertainty of the estimation procedure in order to achieve a more efficient algorithm.

KALMANSAC [VJFS05] was designed for real-time tracking and the estimation of structure from motion. It is derived from pseudo-Bayesian filtering algorithms in a sampling framework and can handle sequences containing large number of outliers. Other examples from robotics are Preemptive RANSAC [Nis03] and Iterative RANSAC [KK06]. One thing they have in common is that the order of the scoring of the pairs of matches is planned in order to avoid scoring useless pairs, i.e. outliers.

MultiRANSAC [MZM05] is a parallel extension of the sequential RANSAC that allows to deal simultaneously with multiple models, which have the advantage of being able to cope with a high percentage of outliers. GASAC [RH06] is another parallel algorithm using a genetic algorithm approach. RANSAC has a low probability to find the correct solution when the data is quasi degenerate and QDEGSAC [FP05] was proposed for use in such cases. NAPSAC[MTN\*02] takes advantage of the fact that if an inlier is found then any point close to that point will have a high probability to be an inlier.

Very little work has focused on maximizing the set of inliers guaranteeing the result to be repeatable, as noted by Li [Li09]. Instead many algorithms focus on the goodness of the fit, such as R-RANSAC, Lo-RANSAC, MLESAC and preemptive RANSAC, just to mention a few. Li proposes a different approach reformulating the problem as a mixed integer pro-

gram, which is solved using a tailored branch-and-bound method. Others [OEK08] have chose to use standard RANSAC with convex programming and a post-validation scheme. The idea is to verify whether the solution is the optimal or if there exists no solution with more than 50% inliers. Yet another approach [EK08], which does not use RANSAC but still finds an optimal solution is tailored for the problem of estimating the position and orientation of a calibrated camera from an image of a known scene.

The above list is by far complete but serves as an overview of some of the more important variants as well as underlining the fact that RANSAC have been investigated and enhanced in many ways for different applications. A performance evaluation of some of the more important variants of RANSAC is done by Choi et al. [CKY09] and a comparative analysis of RANSAC is given by Raguram et al. [RFP08]. Lowe [Low04] proposed to use the Hough transform [DH72] for clustering data, and some hybrids exists [HH07]. Nonetheless, RANSAC remains, with all its variants, the predominant method for finding inliers.

## 2 THE PROPOSED METHOD

The proposed method is based on several observations regarding the set obtained after sampling, estimation of the model and scoring. Algorithm 1: *Optimal-Ransac* shows the main part, which randomly samples the minimal points required in the set of corresponding pairs  $P$ , using Algorithm 2: *randsample*. Moreover, a model  $M$  is estimated using the algorithm *model* and the number of tentative inliers are counted (scored) using the algorithm *score*. These latter algorithms are not specified here but are the main part of any RANSAC algorithm. In our case we used the Matlab<sup>®</sup> algorithms provided by Peter Kovesi on his homepage [Kov]. The main sampling is performed in Algorithm 2: *resample*, whenever the number of tentative inliers  $\eta'$  is larger than 5 for reasons explained in the *Discussion*.

### The Stopping Criterion

A reliable stopping criterion is needed, since in most cases there is no a priori information at hand about how many inliers there are in the set. For tracking applications there is usually some useful guess as the number of tracked points are almost the same for every image. However for a set of two images no such information is available.

For standard RANSAC it is possible to compute the probability of finding more inliers. However, this does not work well for highly contaminated sets as this criterion indicates that more inliers should be found even if all are already found and therefore a better criterion is necessary. Experimentally, it was found that the proposed algorithm will very rarely come to the same consensus twice unless it is the optimal set. This might not

**input** :  $P$  : Set of all points,  $\varepsilon'$  : Tolerance for general sampling,  $\varepsilon$  : Tolerance for the final set of inliers;

**output**:  $M$  : Model for the tentative inliers,  $\mathcal{T}$  : Tentative inliers in  $M$ ,  $\eta$  : Number of tentative inliers;

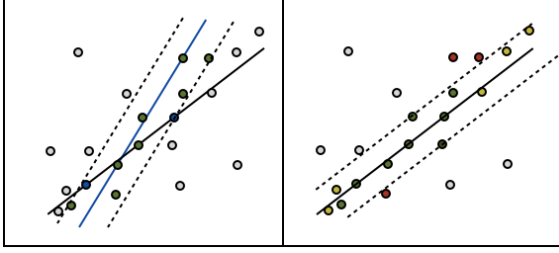
```

 $\sigma \leftarrow 0$ ;
 $\eta \leftarrow 0$ ;
while  $\sigma < 1$  do
     $\tau = \text{randsample}(\eta, 4)$ ;
     $M \leftarrow \text{model}(P(\tau))$ ;
     $\mathcal{T}' \leftarrow \text{score}(M, P, \varepsilon')$ ;
     $\eta' = |\mathcal{T}'|$ ;
    if  $\eta' > 5$  then
         $[M, \mathcal{T}', \eta] = \text{resample}(P, M, \varepsilon, \mathcal{T}', \eta')$ ;
        if  $\varepsilon' < \varepsilon$  then
             $[M, \tau] = \text{pruneset}(M, P(\mathcal{T}'), \varepsilon')$ ;
             $\mathcal{T}' \leftarrow \mathcal{T}'(\tau)$ ;
             $\eta' = |\mathcal{T}'|$ ;
        end
    end
    if  $\eta > 5$  and  $\eta = \eta'$  then
        if  $|\mathcal{T}| = |\mathcal{T}'|$  then
             $\sigma \leftarrow \sigma + 1$ ;
        else
             $nes \leftarrow 0$ ;
             $\mathcal{T} \leftarrow \mathcal{T}'$ ;
        end
    else if  $\eta' > \eta$  then
         $\sigma \leftarrow 0$ ;
         $\eta \leftarrow \eta'$ ;
         $\mathcal{T} \leftarrow \mathcal{T}'$ ;
    else if  $\eta' = \eta - 1$  then
         $\sigma \leftarrow 0$ ;
         $\eta \leftarrow \eta'$ ;
         $\mathcal{T} \leftarrow \mathcal{T}'$ ;
    end
end

```

**Algorithm 1:** OptimalRansac. Find tentative inliers by the resample algorithm. Prune the set to the final tolerance. Stop when the set is equal to the previous set. Also handle the rare cases when one more inliers is found.

be true for all flavors of RANSAC, but the proposed algorithm finds the optimal set very effectively and will therefore usually not come to the same consensus unless it is the optimal set. This works surprisingly well for most sets, however when the set of tentative inliers is very small. i.e. less than 30, the probability of finding a non optimal set twice increases dramatically. A simple solution to this problem is to require more than two equal sets in order to assure that the final set is the optimal one for such cases.



**Figure 1: Left: Two points (blue) within the set of inliers are chosen to construct a new line (black). The tentative inliers (right) now fits the set better and includes also some new points (yellow) and some others are excluded (red).**

One problem that can arise in some rare circumstances is that two different optimal sets are found where  $S_a$  contains one inlier more than set  $S_b$ . If each of these sets are rescored using their transformation model, the very same sets are obtained once again. We found that by pruning the set to a lower tolerance  $\epsilon'$  forced the algorithm to find the same set  $S_b$  in almost every case. Nonetheless, it was necessary to add the last **else if** in Algorithm 1 *OptimalRansac* to handle such rare cases.

### Resampling

Chum et al. [CMK03] [CMO04] proposed to resample the set of tentative inliers already found as the set might contain several inliers but also some outliers that must be removed. This is one of the key parts of their LO-RANSAC algorithm and they perform the iteration 10 times on up to half the current set. We used 8 iterations on a quarter of the set as we got better results using a smaller part of the set and 8 times was enough.

It is easier to understand how RANSAC works for finding lines rather than finding transformations between images. Therefore, an example of line fitting will be used to prove the importance of resampling. Figure 1 (left) shows how two points within the set of tentative inliers are chosen (blue) to construct a new line (black). The new line and tentative inliers are shown to the right. The line fits the set better than before. However, Chum et al. use this in their LO-RANSAC approach only when standard RANSAC scores its best result. Hence, their method tries to optimize this best result, while in Algorithm 2: *resample* it is done whenever more than 5 inliers are found.

### Rescoring

Chum et al. [CMK03] [CMO04] and Lebeda et al. [LMC12] also use rescoring to increase performance and reliability. Nevertheless, they do it in a different way as they propose to rescore the set a fixed number of times while decreasing the tolerance  $\epsilon$  in each step. This will prune the set but also give more inliers. However, by keeping the same tolerance and iterate

```

input :  $P$  : Set of all points,  $M$  : Current model,
         $\epsilon$  : Tolerance,  $\mathcal{T}$  : Tentative inliers in  $P$ ,
         $\eta$  : Number of tentative inliers;
output:  $M$  : New model,  $\mathcal{T}$  : Tentative inliers in
         $P$ ,  $\eta$  : Number of tentative inliers;

 $P' = P(\mathcal{T})$ ;
 $i \leftarrow 0$ ;
while  $i < 8$  do
     $i \leftarrow i + 1$ ;
     $\tau = \text{randsample}(\eta, \max(4, \eta/4))$ ;
     $M' \leftarrow \text{model}(P'(\tau))$ ;
     $\mathcal{T}' \leftarrow \text{score}(M', P', \epsilon)$ ;
    if  $|\mathcal{T}'| > 5$  then
         $[M', \mathcal{T}', \eta'] \leftarrow \text{rescore}(S', \epsilon, \mathcal{T}')$ ;
        if  $\eta' > \eta$  then
             $P' \leftarrow P(\mathcal{T}')$ ;
             $M \leftarrow M'$ ;
             $\eta \leftarrow \eta'$ ;
             $\mathcal{T} \leftarrow \mathcal{T}'$ ;
             $i \leftarrow 0$ ;
        end
    end
end

```

**Algorithm 2: Resample.** Resamples the set of points using up to a quarter of the tentative inliers. If the number of inliers in the whole set using that model is higher than before, then the resulting set is iteratively rescored and if a larger set is obtained the algorithm starts all over.

until the set does not change anymore as in Algorithm 3: *rescore*, the set will soon converge to the optimal set.

We noted that if the set  $S_0$  obtained after scoring is used to compute the transformation again (reestimating) and then rescoring the set, the resulting set  $S_1$  might contain more points than the set  $S_0$ . This procedure can be repeated as long as the set changes. It can nonetheless happen that the set starts to wobble so that set  $S_a$  becomes  $S_b$  after reestimation and rescoring and this set once again becomes  $S_a$  after another reestimation and rescore. As it cannot be assured that this is the only case that can occur and that the wobbling might include more sets, it is necessary to set an upper limit for how many iterations should maximally be performed in order to avoid to get stuck and we chose 20 in our tests. Note that it is not possible to assure that the obtained set is free of outliers. Hence resampling is necessary in order to remove these so that the optimal set can be found, which is free of outliers. However, it can contain points that are a bit off and therefore pruning is necessary as discussed in section 2.

Figure 2 (left) shows how two points in the set are chosen to compute the transformation, which in this case is a line (blue). All points (blue and green)

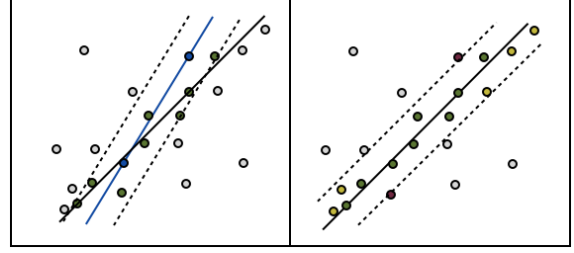
**input** :  $P$  : Set of all points,  $\varepsilon$  : Tolerance,  $\mathcal{T}$  : Tentative inliers in  $P$ ,  $\eta$  : Number of tentative inliers;  
**output**:  $M$  : New model,  $\mathcal{T}'$  : Tentative inliers in  $P$ ,  $\eta'$  : Number of tentative inliers;  
 $j \leftarrow 0$ ;  
 $\eta' \leftarrow |\mathcal{T}|$ ;  
 $\mathcal{T}' \leftarrow \mathcal{T}$ ;  
**while**  $j < 20$  **do**  
   $j \leftarrow j + 1$ ;  
   $M \leftarrow \text{model}(P(\mathcal{T}))$ ;  
   $\mathcal{T} \leftarrow \text{score}(M, P, \varepsilon)$ ;  
   $\eta \leftarrow |\mathcal{T}|$ ;  
  **if**  $\eta > 5$  **then**  
    **if**  $\eta \neq \eta'$  **then**  
       $\eta' \leftarrow \eta$ ;  
       $\mathcal{T}' \leftarrow \mathcal{T}$ ;  
    **else if**  $\mathcal{T}' = \mathcal{T}$  **then**  
       $j \leftarrow 20$ ;  
    **else**  
       $\eta' \leftarrow \eta$ ;  
       $\mathcal{T}' \leftarrow \mathcal{T}$ ;  
    **end**  
  **else**  
     $j \leftarrow 20$ ;  
  **end**  
**end**

**Algorithm 3:** Rescore. Repeatedly reestimates the model and rescores the set until the set does not change anymore. Prevent getting stuck in an eternal loop by not doing more than 20 iterations.

within the tolerance (dotted lines) are tentative inliers. All these points are used to compute the average line passing through them (rescoring), giving the new line (right), which obviously fits the set better. Some points previously considered inliers are now considered outliers (red) and vice versa (yellow). The conclusion is that repeated reestimation and rescoring can be used to fit the set better. Of course, not every set will benefit from reestimation and rescoring but in section 3 it will be shown in a number of tests that the number of iterations needed is generally substantially reduced.

### Pruning

Once a good set of tentative inliers is obtained, where all fall under a certain tolerance  $\varepsilon$ , it is possible to prune the set further by using a tolerance  $\varepsilon'$ , so that  $\varepsilon' < \varepsilon$  and doing rescoring using the already estimated model  $M$ . However, the model  $M$  is estimated using all tentative inliers, also the ones that should be removed. In other words, the model used is not the best one. Hence, when the threshold  $\varepsilon'$  is rather low some true inliers might be removed along with the ones that are a few pixels wrong. A better idea is to remove



**Figure 2:** Left: A line is constructed from two points (blue). All tentative inliers (green and blue) are used to construct a new line (black). The tentative inliers now fits the set better (right) and also includes some new points (yellow) and some others are excluded (red).

just the most extreme inlier, then reestimate the model  $M_i$ , using the remaining inliers in the set  $S_i$ . The index  $i$  denotes the iteration number. This process is repeated until all tentative inliers lie within the tolerance  $\varepsilon'$ . This assures that the best fitting model  $M_i$  is used every time an extreme inlier outside the threshold is removed. Algorithm 4: *Pruneset* performs the necessary pruning.

**input** :  $P'$  : Set of points to prune,  $M$  : Current model for  $P'$ ,  $\varepsilon$  : Tolerance;  
**output**:  $M$  : New model,  $\mathcal{T}'$  : Tentative inliers in  $P'$ ;  
 $\eta \leftarrow |P'|$ ;  
 $\rho \leftarrow 1$ ;  
 $\mathcal{T}' \leftarrow \{1 : \eta\}$ ;  
**while**  $\eta > 5$  **and**  $\rho = 1$  **do**  
   $[\mathcal{T}, \delta] \leftarrow \text{score}(M, P'(\mathcal{T}'), \varepsilon)$ ;  
   $[\varepsilon', \tau] \leftarrow \max(\delta)$ ;  
  **if**  $\varepsilon' > \varepsilon$  **then**  
     $\mathcal{T}'(\tau) \leftarrow \{\}$ ;  
     $M \leftarrow \text{model}(P'(\mathcal{T}'))$ ;  
  **else**  
     $\rho \leftarrow 0$ ;  
  **end**  
**end**

**Algorithm 4:** Pruneset. Prunes the set by removing the most extreme tentative inlier. Recompute the transformation and start all over again. Note that the output  $\mathcal{T}$  from *score* is not used, instead  $\max(\delta)$ , i.e. the max distance, is used to prune  $\mathcal{T}'$ .

## 3 RESULTS

The proposed RANSAC algorithm was executed on several sets of aerial images with a spatial overlap. Furthermore, it was tested on a problem arising in a medical application where the problem is to find the central axis of a bone structure in a volume set. We also tested it on images of houses for panograph stitching, where there are several sub optimal sets due to the



perspective. It will sometimes fail to find the optimal set even if it always finds a set with many inliers, i.e. a sub optimal set.

The Harris corner detector [HS88] was chosen instead of the more accurate SIFT detector [Low04]. The reason is simply because the proposed method is aimed at working also for highly contaminated sets. Since the matching of Harris points being used is less accurate than SIFT it produces more easily such sets. Nonetheless, there is nothing that prevents the use of the proposed method using SIFT feature point sets, or any other feature detector for that matter. An ad hoc setting of the tolerance was used for all images  $im$ , with  $\epsilon' = 1/(2 \min(\text{size}(im)))$  and  $\epsilon = 8\epsilon'$ , which gives a high precision of the accuracy for the final inliers after pruning, while still allowing the transformation to find many inliers in the resampling and rescoring. Generally, a low  $\epsilon$  will make the algorithm run longer, but setting  $\epsilon$  too high will make it fail as outliers will be regarded as inliers.

In order to make a fair comparison to standard RANSAC the iterations reported are computed in the following way. The number of iterations  $\Phi$  in the main loop as well as the time  $\Omega$  is computed for the standard RANSAC operations: sampling, estimating the model and scoring the number of inliers. Then the time  $\Psi$  to compute the proposed Algorithm 2, 3 and 4: *resample*, *rescore* and *prune* as well as handling special cases and the stopping criterion, is computed. The total iterations  $\Upsilon$  is expressed in terms of standard RANSAC iterations. Hence  $\Upsilon = \Phi + (\Phi/\Omega)\Psi$

Table 1 show the results of running the algorithm on four pairs of aerial photos. The measured number of iterations  $\Upsilon$  can be compared to the theoretical number of iterations  $N$  and the speedup (rounded) is computed as  $\Pi = N/\Upsilon$ . The inlier ratio  $\gamma$  is computed (as indicated in the table) as the number of inliers divided by the size of the set. The number of deviant sets, i.e. a set with other inliers than the majority, are indicated by  $\xi$ .

The proposed algorithm was also applied on medical datasets in the form of three 3D computed tomography (CT) images of fractured wrists. The results are shown in Table 2. The underlying task here was to identify the central long axis of the radius bone in the wrist, which can be achieved by (1) segmenting the radius bone and converting it to a surface mesh, (2) selecting a part of the radius shaft located beneath the fracture (top row: a), (3) computing per-vertex normals for the selected surface, (4) mapping the surface normals to points on a unit-sphere (top row: b), and (5) using RANSAC to robustly fit a plane to the dense band formed on the point cloud (top row: c). The normal of that plane will correspond to the direction of the long axis. See [NCM\*12, CG01] for more details. Obtaining precise (sub-degree) measurements of this axis

is of great interest to orthopedic surgeons who need to assess the displacement of wrist fractures in clinical practice or scientific studies.

## 4 DISCUSSION


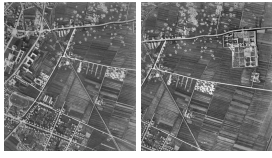
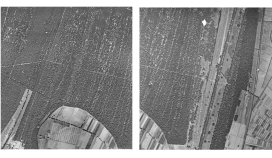
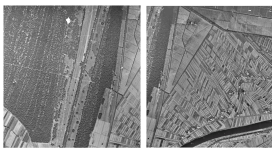
Lebeda et al. [LMC12] propose an improvement of the LO-RANSAC algorithm that is similar but different from our algorithm in a couple of distinctive ways. The similarity is that resampling of the set (Lebeda et al refers to it as the inner RANSAC or the LO step) is done a number of times and each time the set is being rescored using the estimated model obtained from those samples. The differences are the following: they decrease the tolerance for each estimation and scoring, while the algorithm proposed in this paper iterates using the same tolerance until the set does not change anymore, which either means a local maxima is found or that an optimal set is obtained. By decreasing the tolerance, the set is pruned, which is desirable. Nonetheless, it is better to prune once the optimal set is found, since pruning in the rescoring diminishes the possibility to find more inliers as the “search area” marked by the dotted lines in Figure 2 is shrinking. Moreover, if the tolerance is decreased too much there is a chance that inliers that fall under the final tolerance are removed as the transformation is computed using also some inliers above that tolerance.

Another very important difference is that if a larger set is found in Algorithm 2: *resample*, then the process of resampling starts all over using that set instead, since it is larger and will more probably lead to the optimal set. Hence it will make the algorithm come to consensus faster as it will work on a growing set.

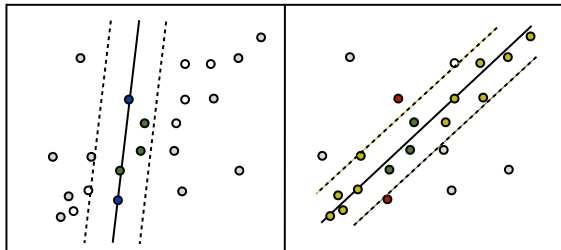
Moreover, Chum et al. propose to do the LO step only if a highest number of inliers is found from ordinary sampling while we propose to perform it whenever more than 5 inliers are found. They have performed their tests on sets with rather high inlier ratios (most of them around 70% or more) and the probability to find a set with many inliers is therefore rather high. Hence, the iterative reestimation and rescoring with pruning is bound to come to the same consensus in each run. However, for low inlier ratios the probability to find a large tentative set is low and if the LO-step happen to find a local maxima it will take many iterations before the standard sampling, estimation and scoring scheme finds a larger set to be optimized in the LO-step.

One fact that makes it important to perform the optimization even on small sets is that even a small contaminated set can lead to consensus. In fact, even if there is only one true inlier in the original samples (4 for the image pairs and 3 for the medical application), the resulting set after estimation and scoring might have more than the number of inliers required to compute the transformation. It seems like nobody



Images	Results			
Pisa, Central				
	$\Upsilon$	791.35	183.23	222.77
	$N$	2969098.7	14471.8	52374.4
	$\Pi$	3752	79.0	235
	$\gamma$	72/1800 = 0.04	33/218 = 0.15	45/410 = 0.11
	$\xi$	0	0	0
Pisa, East				
	$\Upsilon$	828.72	212.47	235.55
	$N$	843.3	49.6	934.6
	$\Pi$	1.02	0.23	3.97
	$\gamma$	554/1800 = 0.31	218/355 = 0.6	240/800 = 0.3
	$\xi$	0	0	0
Pisa, West				
	$\Upsilon$	527.35	141.2	194.26
	$N$	233234.3	2852.8	4978.3
	$\Pi$	442	20.2	25.6
	$\gamma$	136/1800 = 0.08	67/295 = 0.23	117/592 = 0.2
	$\xi$	0	0	0
Pisa West/Arno				
	$\Upsilon$	1346.5	476.55	372.62
	$N$	2269820.7	33555.9	47556.2
	$\Pi$	1686	70.4	128
	$\gamma$	77/1800 = 0.043	33/269 = 0.12	66/587 = 0.11
	$\xi$	0	1	3

**Table 1: ©MiBAC-ICCD, Aerofototeca Nazionale, fondo RAF. The number of iterations (theoretical) and the mean and standard deviation for number of iterations and inliers for different matchings and images.**

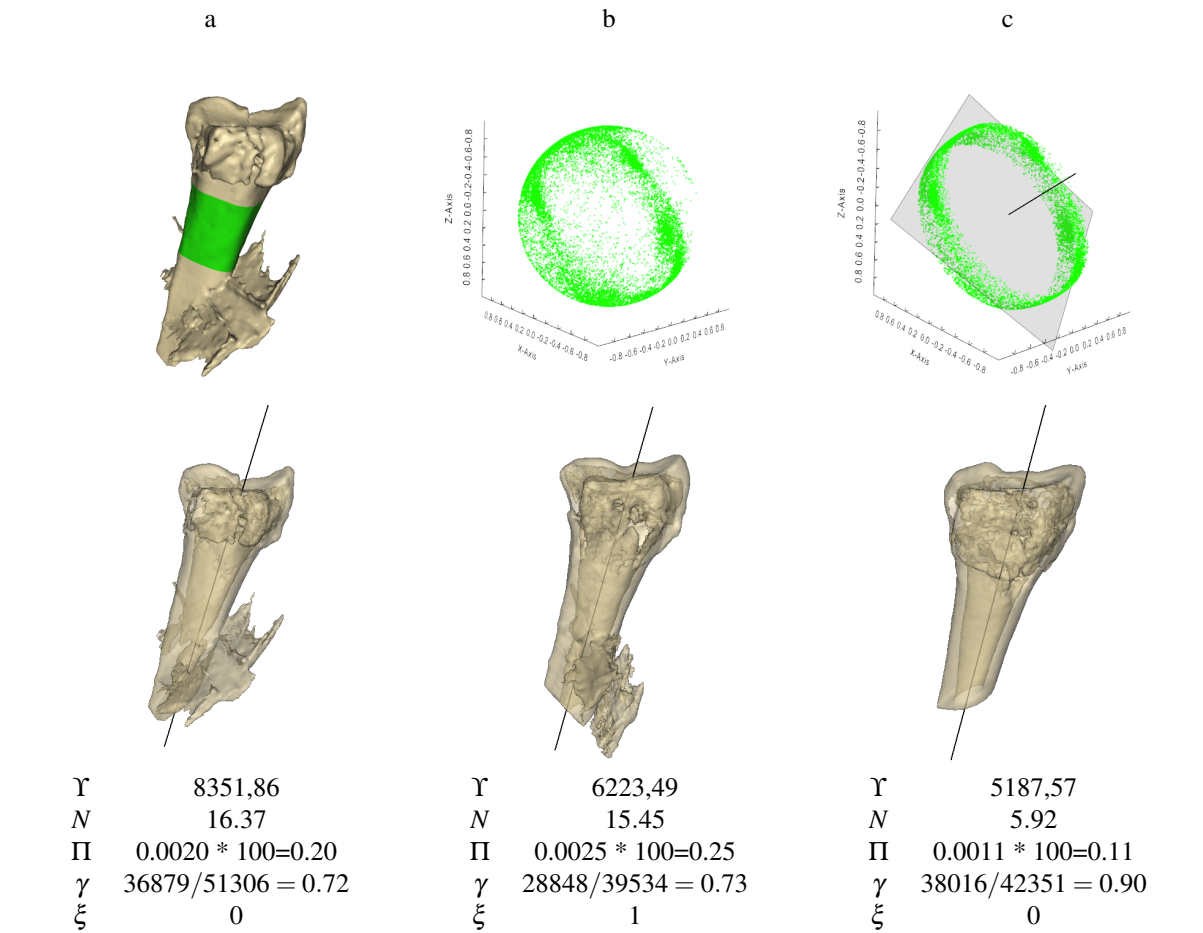


**Figure 3: Left: Two points (blue) within the set are chosen to construct a line (black). Right: The best fitting line contains the same green points and additional yellow points, but the original sampling points (blue) are now considered being outliers and are depicted in red.**




has exploited this fact before, not at least for image pair homography. This finding is very important as it will speed up RANSAC for highly contaminated sets. From the example in Figure 3, one can see that it is possible to find inliers from one estimation and scoring even if the initial samples are all outliers. The tables clearly show that a notable speedup can be gained for highly contaminated sets. The largest speedup (3752) was obtained for the first pair of images in Table 1. One can also note that when the inlier ratio goes up, close

to 50% the speedup decreases to under 1 and the algorithm becomes slower than the theoretical number of iterations. In our case we chose 99.95% number of inliers as we aimed at making the algorithm have less than 5 deviating sets in the 10 000 runs that was performed on each set of images. Different settings for each pair of images was used to produce the tables and this was achieved by changing the number of points obtained from the Harris corner detector and also by changing the lowest response required to be considered a good match in the matching procedure.

The probability to sample one inlier in a set is proportional to the inlier ratio:  $\rho = \eta / |P|$ , where  $\eta$  is number of inliers in the set  $P$ . Hence the probability to sample four inliers is  $\rho^4$ , while the probability to sample one inlier is just  $\rho$ . Assume that the inliers ratio is  $\rho = 0.9$ , then finding one inlier takes  $1/\rho = 1.11$  iterations while  $1/\rho^4 = 1.52$  iterations are needed to find four inliers. This can be compared to the case when the inliers ratio is just  $\rho = 0.1$  then finding one inlier takes  $1/\rho = 10$  iterations while  $1/\rho^4 = 10\,000$  iterations are needed to find four inliers! Nevertheless, it must be kept in mind that not every configuration containing just one inlier will lead to consensus. Therefore, to prove that the reasoning still holds a simple



**Table 2:** Top row from left to right: A section of the bone is selected manually; The surface normals are mapped to points on a sphere; The central axis is found by the Proposed RANSAC method. Bottom row shows results from three different cases, whereof the first one is the case in the top row.

Images	Results			
Bologna				
	$\Upsilon$	531.76	281.27	344.05
	$N$	190810.5	10083.6	18246.0
	$\Pi$	359	35.9	53.0
	$\gamma$	$143/1800 = 0.079$	$56/338 = 0.166$	$115/805 = 0.143$
	$\xi$	0	0	0
Venice				
	$\Upsilon$	774.85	286.72	454.08
	$N$	901293.9	59207.4	240222.3
	$\Pi$	1163	206	529
	$\gamma$	$97/1800 = 0.054$	$38/357 = 0.106$	$60/800 = 0.075$
	$\xi$	5062	0	1
Florence, Ponte Vecchio				
	$\Upsilon$	398.77	165.88	205.12
	$N$	3144.4	146.7	1128.3
	$\Pi$	7.89	0.88	5.5
	$\gamma$	$399/1800 = 0.222$	$210/443 = 0.474$	$229/800 = 0.286$
	$\xi$	0	0	0

**Table 3:** ©Anders Hast. The number of iterations (theoretical) and the mean and standard deviation for number of iterations and inliers for different matchings and images.

test was conducted on the first pair of images in Table 1 and it was recorded how many inliers below the higher threshold  $\varepsilon$  was actually in the set before *resampling* whenever an optimal set eventually was found after *resampling*. On average 2.18 of the four initial samples were inliers and this number varied from 1 to 4. Clearly, consensus can be reached with less than four inlier samples. After scoring 9.06 were considered inliers but only 7.11 actually were inliers, meaning that almost two of these were in fact outliers. On average it took  $\Phi = 47.85$  iterations of the outer loop before an optimal set was found, which can be compared to the theoretical  $N = 2969098.7$  as shown in the table.

There is a limitation in the stopping criterion that must be mentioned. When the inliers are less than about 30 it can happen that the same set is found twice even if it is not the optimal set. Therefore we suggest to increase the parameter  $\sigma$  in Algorithm 1: *Optimal-Ransac* and hence require more than two sets to be equal before the algorithm terminates when less than 30 inliers are found in two equal sets.

#### 4.1 Panographs

Note that in Table 3 on row 2 there is one setting that gives 5062 deviant sets. The reason is that the algorithm finds two different suboptimal sets of inliers and none of them is the optimal set containing *all* true inliers. The image is quite challenging for matching and is constituted by several planes or surfaces. One is found on the water in the canal and the others on the walls etc. The DLT can simply not cover the complex transformation in the image when the tolerance is set very low. The result is that the two suboptimal sets will contain a majority of inliers that are in common, but some points that are unique for each set. It is possible to find a set containing all inliers by increasing the tolerance but then there is a risk that the set will contain correspondences that are one or more pixels wrong and should be considered outliers. This is a common problem for RANSAC and can be solved by removing the inliers from the set when consensus is reached and repeat the whole process with the remaining set. However, determining which set of inliers are on the water and which one is on the walls etc is out of scope for this paper. Hence, it cannot be assured that the algorithm finds the optimal set or whether a sub optimal set was found for such pictures as shown in Table 3. Obviously this is a drawback with the algorithm and in fact is a drawback for most flavors of RANSAC. Nevertheless, for the images in Table 1 the ground is far from the viewer and can be considered being all in the same plane.

#### 4.2 Medical datasets

In contrast to the aerial images, the inlier ratio in these datasets is significantly higher, usually 50% or more. However, since the selected part of the bone is slightly

flattened and conical rather than cylindrical, standard RANSAC tends to get stuck in (and wobble between) local minima in each run and produce axes that are not equally well-aligned to all sides of the bone. The method described in [NCM\*12] addressed this issue by first running RANSAC multiple times to generate a set of candidate axes, and then selecting the mean of these axes as the true axis. Although that method turned out to be precise enough for the intended application, the proposed method will find the optimal axis directly. In Table 2, the speedup is multiplied with 100 as ordinary RANSAC needed to be run 100 times in order to give a reliable result. Hence, our proposed method is 5 to 10 times slower but the result is more reliable as the very same set is obtained in each run.

Table 2 show the results of running our algorithm 1000 times each on the three medical datasets. The parameter settings for this experiment were  $\varepsilon = 0.275$  and  $\varepsilon' = 0.25$ .

### 5 CONCLUSION

We have proposed an algorithm that is similar to LO-RANSAC as it resamples the set of tentative inliers a number of times and then performs iterative estimation of the model and scores the number of inliers. It is important to notice that the proposed algorithm is different from the LO-RANSAC in the following ways:

- The optimization is performed whenever a set with more than five tentative inliers is found. This is important for sets with a low inlier ratio.
- Whenever a larger set is found in the resampling step, the resampling starts all over with that set so it will work on a growing set until the largest set is found with the help of iterative reestimation and resampling.
- The iterative reestimation and resampling uses the same tolerance so that the set will grow as much as possible. The iteration does not stop until the set stops changing, which means that there is a high probability that an optimal set is found.
- Pruning is done afterwards with a lower tolerance so that only the very best inliers are kept. The transformation is recomputed using the remaining inliers in each step.

The main drawback with the algorithm is that when the image contains more than one plane, such as the images in Table 3 then it cannot be assured that the optimal set is found as there can be several sub optimal sets that fulfill the transformation with a given tolerance. Nevertheless, this is a problem for most versions of RANSAC and we did not try to solve it in this paper. In any case, the algorithm will find a suboptimal set efficiently, especially for low inlier ratios, but

there is unfortunately no guarantee that the algorithm will find the same suboptimal set in every run.

The proposed improvements for RANSAC generally yield an optimal set in more than 99.95% of the cases for aerial images. LO-RANSAC will perform close to the theoretical number of iterations in equation 1 and thus be faster for high inlier ratio sets. The proposed algorithm will on the other hand have the advantage of a substantial speedup for highly contaminated sets and will handle sets with even just 4% of inliers. Another advantage is that it will find the optimal set in each run for both aerial images and for finding planes in the medical application it was tested on.

## 6 ACKNOWLEDGEMENTS

This work was partly carried out during the tenure of an ERCIM "Alain Bensuossan" Fellowship Programme. An implementation in Matlab® is found at: [www.cb.uu.se/~aht/code.html](http://www.cb.uu.se/~aht/code.html)

## 7 REFERENCES

- [BL07] Brown M., Lowe D. G.: Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision* 74, 1 (2007), 59–73.
- [CG01] Chaperon T., Goulette F.: Extracting Cylinders in Full 3D Data Using a Random Sampling Method and the Gaussian Image. In *Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), VMV '01, AKA-Verlag, pp. 35–42.
- [CKY09] Choi S., Kim T., Yu W.: Performance evaluation of ransac family. In *British Machine Vision Conference* (2009), pp. 1–12.
- [CM05] Chum O., Matas J.: Matching with prosac - progressive sample consensus. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2005), pp. 220–226.
- [CM08] Chum O., Matas J.: Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 8 (2008), 1472–1482.
- [CMK03] Chum O., Matas J., Kittler J.: Locally optimized ransac. In *the Annual Pattern Recognition Symposium of the German Association for Pattern Recognition* (2003), pp. 236–243.
- [CMO04] Chum O., Matas J., Obdrzalek S.: Enhancing ransac by generalized model optimization. In *Asian Conference on Computer Vision* (2004).
- [DH72] Duda R. O., Hart P. E.: Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM* 15 (1972), 11–15.
- [EK08] Enqvist O., Kahl F.: Robust optimal pose estimation. In *European Conference on Computer Vision* (2008).
- [FB81] Fischler M. A., Bolles R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24 (1981), 381–395.
- [FP05] Frahm J. M., Pollefeys M.: Ransac for (quasi-) de-generate data (qdegsac). In *IEEE Conference on Computer Vision and Pattern Recognition* (2005), pp. 220–226.
- [HH07] Hollander R. J. M. D., Hanjalic A.: A combined ransac-hough transform algorithm for fundamental matrix estimation. In *British Machine Vision Conference* (2007).
- [HS88] Harris C., Stephens M.: A combined corner and edge detection. In *Alvey Vision Conference* (1988), pp. 147–151.
- [HZ03] Hartley R. I., Zisserman A.: *Multiple View Geometry, 2nd edition*. Cambridge University Press, 2003, pp. 32–33, 87–90, 121–122.
- [KK06] K. K. T., Kondo E.: Incremental ransac for online relocation in large dynamic environments. In *IEEE International Conference on Robotics and Automation* (2006), pp. 1025–1030.
- [Kov] Kovsi P.: Matlab and octave functions for computer vision and image processing. <http://www.csse.uwa.edu.au/~pk/research/matlabfns/>.
- [Li09] Li H.: Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *International Conference on Computer Vision (ICCV)* (2009), pp. 1074–1080.
- [LMC12] Lebeda K., Matas J., Chum O.: Fixing the locally optimized ransac. In *Proceedings of the British Machine Vision Conference* (2012), BMVA Press, pp. 95.1–95.11.
- [Low04] Lowe D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (2004), 91–110.
- [MTN\*02] Myatt D., Torr P., Nasuto S., Bishop J., Craddock R.: Napsac: High noise, high dimensional robust estimation - its in the bag. In *British Machine Vision Conference* (2002), vol. 2, pp. 458–467.
- [MvHK\*06] Michaelsen E., von Hansen W., Kirchhof M., Meidow J., Stilla U.: Estimating the essential matrix: Goodsac versus ransac. In *Photogrammetric Computer Vision* (2006), pp. 1–6.
- [MZM05] M. Zuliani C. K., Manjunath B.: The multiransac algorithm and its application to detect planar homographies. In *The International Conference on Image Processing* (2005), vol. 3, pp. 153–156.
- [NCM\*12] Nysjö J., Christersson A., Malmberg F., Sintorn I.-M., Nyström I.: Towards User-Guided Quantitative Evaluation of Wrist Fractures in CT Images. In *Computer Vision and Graphics* (2012), Bolc L., Tadeusiewicz R., Chmielewski L., Wojciechowski K., (Eds.), vol. 7594 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 204–211.
- [Nis03] Nister D.: Preemptive ransac for live structure and motion estimation. In *International Conference on Computer Vision (ICCV)* (2003), pp. 109–206.
- [OEK08] Olsson C., Enqvist O., Kahl F.: A polynomial-time bound for matching and registration with outliers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).
- [RFP08] Raguram R., Frahm J.-M., Pollefeys M.: A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision* (2008), pp. 500–513.
- [RFP09] Raguram R., Frahm J.-M., Pollefeys M.: Exploiting uncertainty in random sample consensus. In *International Conference on Computer Vision (ICCV)* (2009), pp. 2074–2081.
- [RH06] Rodehorst V., Hellwich O.: Genetic algorithm sample consensus (gasac) - a parallel strategy for robust parameter estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop* (2006), pp. 1–8.
- [SLK09] Sattler T., Leibe B., Kobbelt L.: Scramsac: Improving ransac's efficiency with a spatial consistency filter. In *International Conference on Computer Vision (ICCV)* (2009), pp. 2090–2097.
- [TZ00] Torr P. H. S., Zisserman A.: Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding* 78 (2000), 138–156.
- [VJF05] Vedaldi A., Jin H., Favaro P., Soatto S.: Kalmansac: Robust filtering by consensus. In *International Conference on Computer Vision (ICCV)* (2005), pp. 633–640.
- [VL01] Vincent E., Laganieri R.: Detecting planar homographies in an image pair. *Image and Signal Processing and Analysis* (2001), 182–187.
- [Zul09] Zuliani M.: Ransac for dummies. <http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf>, 2009.

# LightCluster - Clustering Lights to Accelerate Shadow Computation

Daniel Wiesenhütter

Andreas Klein

Alfred Nischwitz

Munich University of Applied Sciences  
Lothstrasse 64

80335 Munich, Germany

daniel@wiesenhuetter.me, andreas.klein@hm.edu, nischwitz@cs.hm.edu

## ABSTRACT

In this paper, we propose a method to reduce the amount of shadow maps required for rendering shadows in scenes with many lights. Our idea is to use the spatial relationship of lights to find clusters and replace the lights of a cluster with a single area light. We use a soft shadow algorithm for area lights to approximate the shadows for the clusters. By carefully placing the cluster centers, we can minimize the errors in the shadows. While the clustering only adds a small overhead in the worst case, it can efficiently reduce the number of shadow maps. Thus, in many cases the resulting error in shadows is acceptable compared to the increase in rendering performance.

## Keywords

Light Clustering, Shadow Mapping, Soft Shadows, Shadows for Many Lights.

## 1 INTRODUCTION

Shadows are an important part of a visualization and give the viewer supplemental details about the appearance of objects. In real-time rendering, shadow mapping [Wil78a] is a popular approach to compute shadows. However, a shadow map must be computed for each light and thus, the memory and the computation time increases with the number of lights.

In this paper, we present an approach, called LightCluster, to automatically select representative light sources and accelerate the computation of direct shadows for scenes with many lights. We carefully select light sources as cluster centers and cluster the remaining lights using a minimum distance metric [Wol57a]. We represent each cluster by an area light source and use a soft shadow algorithm to render shadows for each cluster, such as Percentage Closer Filtering [Ree87a] and Percentage Closer Soft Shadows [Fer05a].

In our implementation, we use omnidirectional point lights. However, the approach can be adapted for other light types, such as directional or spot lights. Figure 1 shows an example of our approach.

The main contributions of our paper are:

- A clustering strategy applied to point lights that selects a variable number of existing light sources as cluster centers.
- A minimum distance metric in order to minimize the amount of shadow maps and to trade off between quality and performance.

- An approximation of point light shadows by using an area light source where the area depends on the minimum distance between the clusters.

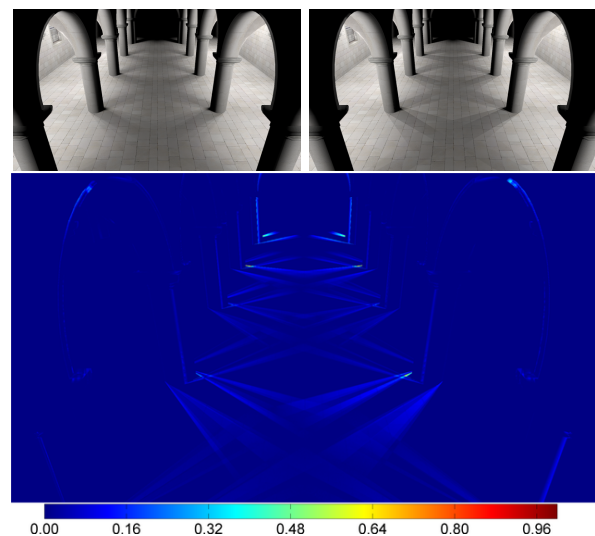


Figure 1: Comparison between a reference solution and our solution for the Dabrovic Sponza scene at 1920x1080 resolution. The top left image shows the reference solution where 80 point light shadows are rendered in 97.8 ms. The top right image demonstrates our solution with 26 cluster shadows using PCSS in 43.1 ms. The bottom image shows the difference between both above images. Note that a dark blue color indicates a low error and red color a high error.



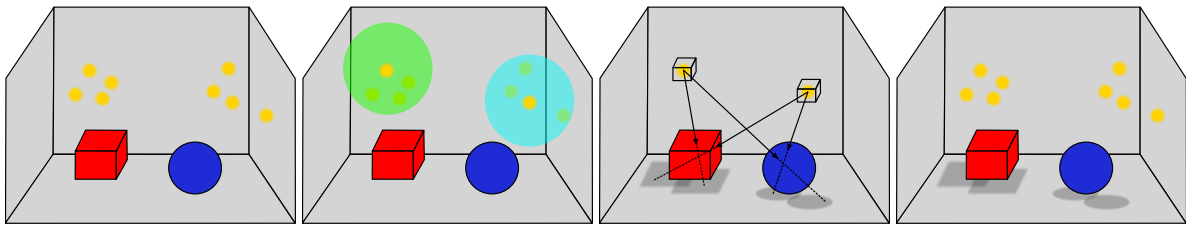


Figure 2: Overview for LightCluster. For a given distribution of point light sources, clusters are selected with the min-distance-cluster algorithm. For each cluster, a cube shadow map is rendered, soft shadows are computed and stored in a visibility texture. This textures are then used during shading to determine the visibility.

## 2 RELATED WORK

### 2.1 Real-Time Soft Shadows

The rendering of soft shadows is an active field of research. Therefore, we focus our review on publications closely related to our work. For an exhaustive survey on other methods see [Eis11a].

Shadow Mapping [Wil78a] is a popular method to compute shadows in real-time rendering. The idea is to assume point light sources and to replace the visibility test by comparing depth values from the light's point of view and the observer's point of view. In order to compute shadows for omnidirectional lights, a cube shadow map can be rendered [Ger04a].

Percentage Closer Filtering (PCF) [Ree87a] computes filtered hard shadows by making multiple shadow comparisons within a filter window. This idea is further extended by Fernando [Fer05a] with Percentage Closer Soft Shadows (PCSS) to realize shadows with variable sized penumbras. Instead of using a fixed filter per pixel, the filter window is scaled according to a penumbra size. The penumbra size can be estimated by calculating an average blocker depth and using similar triangles.

### 2.2 Many-Light Methods

Instant Radiosity [Kel97a] approximates global illumination by distributing virtual point lights (VPLs) and calculating a local illumination model for each VPL. Indirect shadows are realized by rendering a shadow map for each VPL. In order to distribute VPLs by textures, Reflective Shadow Maps (RSMs) [Dac05a] are rendered from the position of light.

In Lightcuts [Wal05a] a binary tree for light sources is built. In each frame the light tree is traversed and a cut is calculated. This light cut determines the relevant lights or representatives with the help of visual criteria, such as geometric properties or material.

Recent approaches try to accelerate the computation of Instant Radiosity by reusing shadow maps for VPLs [Lai07a] or by rendering low resolution shadow maps for a coarse representation of the scene [Rit08a].

Hašan et al. [Haš07a] interpreted the relationship between  $m$  surfaces and  $n$  lights as a  $m \cdot n$  matrix and samples only a small number of rows and columns. In order to minimize flickering in many-light animations, the matrix can be extended to a tensor [Haš08a].

Clustered Visibility [Don09a] accelerates the computation of indirect shadows with RSMs. The method uses k-means clustering on the RSMs to build clusters of VPLs. They interpret each cluster as an area light in order to accelerate the visibility test.

The idea of [Don09a] is closely related to our work. In contrast to Clustered Visibility we focus our work on high frequency shadows for direct lighting. As the lights are not distributed by RSMs, a k-means clustering may lead to errors. A central cluster position can result in a representative light source that is occluded by geometry, i.e. located within walls. Our approach uses an existing light source as a cluster center and clusters the remaining lights with a minimum distance metric.

## 3 LIGHTCLUSTER

LightCluster reduces the total amount of shadow maps in order to minimize the computation of direct shadows for point lights. To describe our approach in more detail, we separate it into two steps. First, the point lights are clustered and cluster centers are selected. Second, for each cluster a shadow maps is rendered, soft shadows are computed and the result is stored in a set of visibility textures. During shading, we select the representative visibility texture for each point light and use it to determine the visibility factor. Figure 2 illustrates the approach.

### 3.1 Clustering

In order to reduce the error in shadows, we perform a two pass clustering with different metrics in each pass. Our clustering proceeds as follows. We first select point lights as cluster centers by using the light range and a minimum-distance metric, which is scaled by the camera distance. This allows us to generate smaller clusters and thus, more shadow maps, near the camera position. In the second clustering pass, the remaining point lights are assigned to the nearest cluster centers. Therefore, the error in shadows due to the reduced amount of

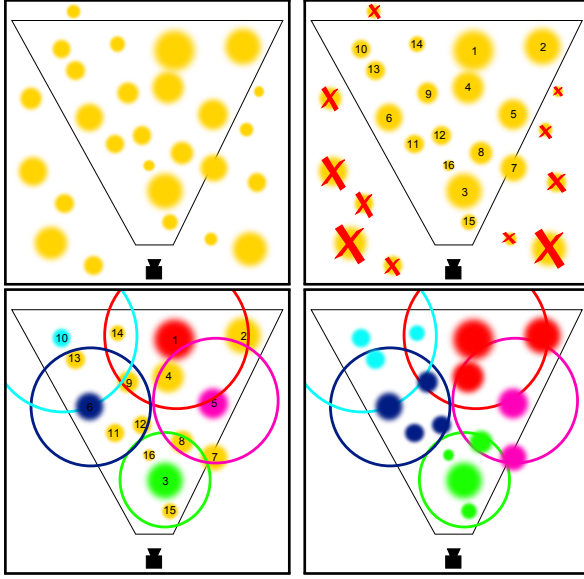


Figure 3: Minimum-distance-cluster algorithm in a 2D example scene. The size of the point lights describes its attenuation. **Top left:** Light distribution in the scene. **Top right:** The point lights are culled against the view frustum and sorted by the attenuation factor. **Bottom left:** Finding cluster centers. Note how the minimum distance  $d_{min}$  increases based on the distance to the camera. Due to the sorting, the lights with the highest range are first tested as cluster centers. **Bottom right:** Assigning the remaining point lights to the closest cluster centers. Without this step, lights would assigned to the first cluster within  $d_{min}$ .

shadow maps can be reduced. Figure 3 illustrates the clustering for a 2D example scene.

### 3.1.1 Minimum-Distance-Cluster Algorithm

The minimum-distance-cluster algorithm [Wol57a] is a hierarchical clustering algorithm and clusters all objects within a user defined minimum distance. The algorithm can be realized in a top-down strategy and proceeds as follows. First, it is assumed that all objects are within a single cluster. The algorithm iterates over all objects of a cluster and splits them into new clusters, if their distance exceeds the defined minimum distance. This step is repeated until no more clusters can be split.

In contrast to a centroid-based clustering, such as k-means [McQ67a], the number of clusters need not be known in advance. Furthermore, the cluster center is an actual light source and no computed centroid position. Thereby, the position for one point light in a cluster is correct for the shadow computation with PCF or PCSS. The error in the shadows for the remaining lights can be adjusted by the minimum distance parameter.

### 3.1.2 Selecting Cluster Centers

In order to find potential cluster centers, we first search for point light sources that will potentially cast a

shadow in the view frustum. As we assume point light sources, we create a bounding sphere for each point light where the radius equals the attenuation range. We then cull the bounding spheres against the view frustum and remove all lights that are classified as outside. In the next step, we sort the visible lights by the attenuation range. As a result, our clustering algorithm will first test the lights with the highest range as cluster centers. This heuristic could be further improved by determining how much of the light source is visible in the view frustum, for example by utilizing occlusion culling or a hierarchical z-buffer [Gre93a].

In the next step, we select the cluster centers. We assume that all lights are within a single cluster and the light with the highest range is the first cluster center. For every point light, we compute the distance to the cluster center and compare it against the minimum distance  $d_{min}$ . In order to generate smaller clusters near the camera position, we scale the minimum distance linearly based on the distance to the camera. The minimum distance can be calculated as follows:

$$d_{min} = d_{init} \cdot \left( 1 - \frac{|C_{pos}|}{z_{far}} \right)$$

where  $d_{init}$  is the initial distance between clusters,  $C_{pos}$  the position of the cluster center and  $z_{far}$  the maximum view distance.

If the distance to the cluster is greater than the calculated  $d_{min}$ , the light is promoted to a new cluster center. The lights that are within  $d_{min}$ , are not automatically assigned to the cluster, but marked as unclustered for the next step.

### 3.1.3 Assigning the remaining Lights

After all cluster centers are selected, we assign the lights to the clusters. We iterate over all lights that are marked as unclustered and calculate the distance to all cluster centers. The light is assigned to the cluster with the shortest distance. In addition, the maximum attenuation range  $C_{att}$  of the cluster center is adapted as follows:

$$C_{att} = \max(C_{att}, d_C + L_{att})$$

where  $d_C$  is the distance to the cluster and  $L_{att}$  is the attenuation of the light source. We use the maximum attenuation range  $C_{att}$  to accelerate the computation of the shadows for a cluster.

## 3.2 Computing Shadows

Instead of rendering shadows for each point light in a scene, we calculate the shadows only for a smaller set of cluster centers. In order to reduce the error in the resulting shadows, we interpret each cluster center as an area light source and calculate the visibility factor using a soft shadow algorithm.

### 3.2.1 Direct Lighting Equation with Light Clusters

Following the notation of [Eis11a], the direct-lighting equation for  $n$  point lights can be expressed as:

$$L_o(p, \omega) = \sum_{i=0}^n L_d(p, \omega, l_i) \cdot L_{c_i} \cdot V(p, l_i)$$

with

$$L_d(p, \omega, l_i) = f_r(p, \omega, p \rightarrow l_i) G(p, l_i)$$

where  $p$  is a point on a surface,  $\omega$  a direction,  $f_r$  is a Bidirectional Reflectance Distribution Function (BRDF),  $V$  the visibility function,  $L_{c_i}$  the color of the  $i$ -th light and  $l_i$  the position of the  $i$ -th point light. The notation  $p \rightarrow q$  is defined as  $\frac{q-p}{\|q-p\|}$ . The geometric term  $G$  is defined as

$$G(p, q) = \frac{\cos(n_p, p \rightarrow q) \cos(n_q, q \rightarrow p)}{\|p - q\|^2}$$

with the normal  $n$ .

With the clustering, we replace the binary visibility of point lights with the visibility of an area light source  $A$  for a cluster  $c$ , which yields the following equation:

$$L_o(p, \omega) \approx \sum_{i=0}^n L_d(p, \omega, l_i) \cdot L_{c_i} \cdot \frac{1}{A_c} \int_{A_c} V(p, l) dl$$

### 3.2.2 Rendering Shadows

We render a cube shadow map for each cluster and interpret the cluster as a disc-shaped area light source. The radius of the area light source is given by the minimum distance  $d_{min}$  of the cluster. We use this radius to scale the filter window of PCSS [Fer05a] and PCF [Ree87a]. Due to the maximum attenuation range  $C_{att}$  of a cluster, we discard a pixel if the distance from the pixel to the cluster center exceeds the attenuation range. The visibility factor is then stored in a texture for each cluster and is accessed during shading. This allows us to calculate the shadows iteratively and reduces the texture memory from a cube shadow map to a screen sized texture per cluster. In this way, we sample the cube shadow maps only once for each cluster and avoid additional PCF or PCSS sampling for each light source during shading.

After all cluster shadows have been computed, we use the set of visibility textures for shading. We shade the scene with each point light source and modulate the resulting color with the visibility value stored in the texture for the point light's cluster.

## 4 IMPLEMENTATION

We implemented our algorithm using Direct3D 11 and Tiled Deferred Shading [Lau10a]. However, LightCluster can also be implemented with any other method for

shading many point lights, such as Tiled Forward Shading [Ols11a]. In each frame, we cluster the point lights, render the G-Buffer, compute the shadows for the clusters and perform the shading.

Our minimum-distance-cluster algorithm is entirely implemented on CPU, as it adds only a small overhead (see section 5 for details). The cluster position, attenuation range and a cluster index per light source is stored in an unordered access view on the GPU and can be accessed in the shaders.

In order to render a cube shadow map in a single render pass, we use a geometry shader to replicate the geometry for each face direction. We store the world-space distance in each face and reconstruct the world-space position from the depth stored in the G-Buffer during the shadow test. The set of visibility textures is realized with a texture array.

## 5 RESULTS

The following results are rendered on an Intel Xeon E5620 CPU with 2.4 GHz, 8 GB RAM and a NVIDIA GeForce GTX 680 graphics card with 2048 MB memory. The screen resolution for all images is 1920x1080 and the resolution of one face in a cube shadow map is set to 1024x1024. We use a Poisson disk with eight samples for PCF and PCSS.

### 5.1 Scenes

We present images and measurements for three different scenes. In the Dabrovic Sponza scene 80 lights are placed in circles of eight lights sideways in the arcade to the right and left (Figure 7a). Within the Cornell Box scene 32 light sources are randomly distributed in the room (Figure 8a). The last scene is a part of a restaurant with two tables and a chandelier. Fourteen lights overall are placed, at the candle stands on the table, at lamps at the wall and at the chandelier (Figure 9a).

### 5.2 Reference solution

As a reference solution to our approach, we compute shadows with cube shadow mapping for every light source. Since shadows are computed for point light sources, a binary visibility test determines if the pixel is in shadow or not. Consequently, 80 cube shadow maps for Dabrovic Sponza (Figure 7b), 32 cube shadow maps for Cornell Box (Figure 8b) and fourteen cube shadow maps are rendered for the Restaurant scene (Figure 9b).

### 5.3 Memory

Compared to the reference solution, LightCluster uses additional memory. During the shadow computation, the cluster visibility values are stored in a texture array. Therefore, the additional memory size is cluster count multiplied by the screen resolution. For example, in the Dabrovic Sponza scene with 26 clusters the additional memory size is 205 MB. In contrast, the memory size for 26 cube shadow maps is 624 MB.



Sponza		Time (ms)		Cornell Box		Time (ms)		Restaurant		Time (ms)	
$d_{init}$	Clusters	PCF	PCSS	$d_{init}$	Clusters	PCF	PCSS	$d_{init}$	Clusters	PCF	PCSS
0.1	80	100.1	115.3	0.3	13	12.9	18.0	0.2	12	36.0	40.1
0.3	44	57.0	67.3	0.4	6	9.8	12.9	0.3	9	28.9	32.9
0.5	26	36.4	43.1	0.5	5	9.1	12.1	0.4	8	26.8	30.5
1.6	10	18.7	20.7	0.6	4	8.6	11.1	0.5	7	24.9	27.8
Ref. with 80 lights		97.8		Ref. with 32 lights		20.0		Ref. with 14 lights		39.3	

Table 1: Duration of LightCluster for the scenes with different  $d_{init}$  factors. The higher the factor, the coarser the clustering and lesser the amount of clusters. Thus, the chosen  $d_{init}$  factor determines the performance.

## 5.4 Performance

We compare our performance results against the reference solution. In all our test scenes we use a maximum view distance  $z_{far}$  of 30 and vary only the initial distance  $d_{init}$  between clusters for the clustering.

As can be seen from Table 1, LightCluster is faster than the reference solution, depending on the chosen  $d_{init}$  and the resulting amount of clusters. The higher the  $d_{init}$  factor, the coarser the clustering and a lesser amount of cube shadow maps is required. Thus, the chosen  $d_{init}$  factor determines the performance of our approach. The worst case scenario for our approach occurs when no lights can be clustered. This can occur when the parameter  $d_{init}$  is chosen too small for a given scene. Table 1 shows this case for the Dabrovic Sponza scene with 80 clusters. The performance in this worst case is slightly slower than for the reference solution, because an additional clustering as well as a PCF or PCSS sampling is performed. However, the worst case can be avoided by comparing the amount of lights with the cluster count and choosing the reference solution render path.

Sponza		Time (ms)		Percentage	
Step		PCF	PCSS	PCF	PCSS
Clustering		0.05	0.05	<0.1%	<0.1%
GBuffer		0.53	0.53	0.9%	0.8%
Shadows		49.20	59.42	86.3%	88.4%
Lighting		7.05	7.05	12.3%	10.4%
Compositing		0.20	0.20	0.4%	0.3%
Total		57.03	67.25	100%	100%

Table 2: Duration of the algorithm steps using the Dabrovic Sponza dataset with 44 clusters.

The duration of the single steps in our algorithm is shown in Table 2 for the Dabrovic Sponza scene with 44 clusters. It can be observed, that the bottleneck in our algorithm is the computation of shadows. The clustering step requires less than 0.1% of the total performance. Thus, a clustering of the light sources can be performed for every frame in order to reduce the amount of cube shadow maps.

Figure 4 shows the performance of the clustering step when using a different amount of lights. The clustering time increases nearly linear with the number of lights.

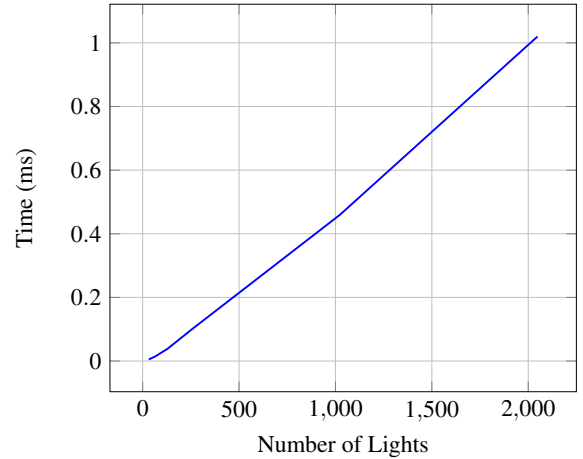


Figure 4: Timings for the min-distance-clustering for a different number of lights.

With 2048 lights, the clustering can be still performed within 1 ms of time.

## 5.5 Error

In order to compare the error of our approach to the reference solution, we use difference images. The dark blue color of a difference image indicates a low error and a red color indicates a high error.

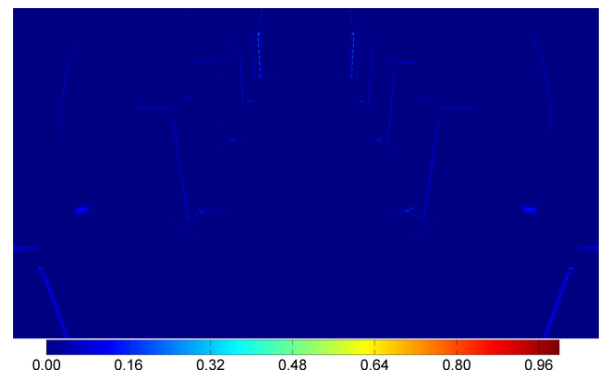


Figure 5: Comparison between the shadows of PCF and PCSS.

Figure 5 shows a difference image between PCSS and PCF shadows. It can be observed, that the PCSS solution generates a contact hardening shadow at the bottom of the columns. Therefore, PCSS provides a more

realistic solution. Nevertheless, PCF can be used to accelerate the computation of shadows further (see Table 1). For the following comparisons we use LightCluster with PCSS.

Figure 7 shows our algorithm with two different cluster settings for the Dabrovic Sponza scene. Figure 7a and 7b shows the light distribution and the reference solution. In the first case, we use an initial distance  $d_{init} = 0.5$  which results in 26 clusters. The cluster locations are illustrated in Figure 7c. As can be seen in the difference image, there is a small error at the shadow boundaries and slightly larger errors near the far wall, where the number of clusters is reduced due to the distance scaling. For the second case, we use an initial distance  $d_{init} = 1.6$ . The clustering step produces ten clusters (Figure 7d). In this case the shadows are rendered for only one out of eight light sources and errors can be clearly seen in the resulting image (Figure 7f) as well as the difference image (Figure 7g). Nevertheless, the general impression of the shadows in the scene is preserved and the rendering performance is increased by a factor of 4.8.

The results in the Cornell Box scene with randomly distributed point lights are presented in Figure 8. The light distribution is shown in Figure 8a and the reference solution in Figure 8b. On the left side, Figure 8c displays the clusters locations with  $d_{init} = 0.5$ , which results in five clusters. Due to the reduction from 32 point lights to five clusters, the image shows errors in the shadows cast from the left cube and below the right sphere (Figure 8e and 8g). By increasing the initial distance to  $d_{init} = 0.6$ , 32 point lights are reduced to four clusters. The results in Figure 8f and 8h shows, that the error on the wall behind the sphere increases.

In the Restaurant scene, fourteen lights are distributed (Figure 9a) and Figure 9b shows the reference solution. Due to the large distance between the light sources in this scene, only the lights of the chandelier and the candles are suited for clustering. In Figure 9c the lights of the chandelier are represented by three clusters, which is accomplished by  $d_{init} = 0.3$ . Figure 9e presents our approximation. The resulting difference image in Figure 9g shows small errors in the shadows cast by the chandelier. By using a initial distance of  $d_{init} = 0.4$ , the light sources of the chandelier can be represented by two clusters (Figure 9d). The results in Figure 9f and 9h show, that the error in the shadows at the floor is increased.

Figure 6 shows the root mean square error (RMSE) for our approach using different initial cluster distances and PCSS in our test scenes. The error increases when the number of cluster centers and thus the number of shadow maps is adapted, e.g. in the Restaurant scene with a distance of 0.4 and 0.5. On the other hand, the error is nearly constant when the lights are assigned to

different clusters according to the second step in the clustering, e.g. in the Cornell scene with a cluster distance between 0.4 and 0.7.

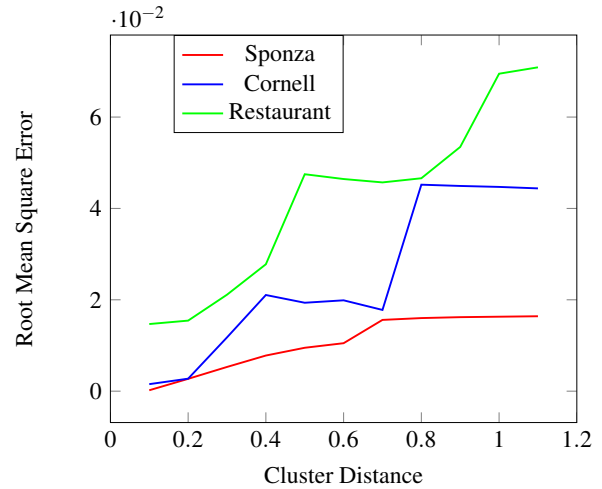


Figure 6: RMSE for different initial cluster distances.

## 5.6 Discussion

We used a minimum-distance clustering algorithm to cluster the light sources. Compared to a k-means clustering, the minimum-distance clustering has the following advantages in this scenario. First, the number of cluster centers must not be known in advance. This simplifies the algorithm and no additional heuristics for estimating the number of cluster centers in advance are required. Second, we place a cluster center always on an existing light source. Therefore, the light position for shadow mapping is always correct for at least one light source in the cluster. The k-means algorithm on the other hand uses the mean of the light positions within a cluster. As the lights are distributed in space, this could result in cluster centers which are located within geometry, e.g. in a wall. Dong et al. [Don09a] solves this for virtual point lights by using the surface normals as an additional cluster criterion. However, this is not possible in our approach, as the light sources can be freely placed in space. Therefore, an additional visibility test for the cluster centers would be required to ensure that they are not located within geometry.

Our approach can introduce temporal artifacts, such as flickering shadows, when the camera is moved or the light sources are animated. These artifacts can be reduced by disabling the view-adaptive scaling in the clustering step and by creating cluster centers for the animated light sources.

The results showed several insights about LightCluster concerning the scalability, quality and overall performance. By adjusting the clustering parameters, a well defined ratio between quality and performance can be chosen. Due to the approximation of the shadows from many lights with a soft shadow from a representative

light source, errors in the shadow can be reduced. As the clustering step is fast, it can be performed in each frame and only adds a small overhead. If light sources can be merged to clusters, LightCluster can be used to increase the rendering performance while maintaining an acceptable shadow quality in many cases.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an approach, called LightCluster, to find representative light sources for approximating the shadows of many-lights. By focusing on high frequency shadows for direct lighting, we introduced a modified min-distance-cluster algorithm that combines different metrics to find a variable number of clusters in the view frustum. These clusters are then interpreted as an area light source which is scaled according to the cluster distance. In order to reduce the resulting errors in the shadows, we compute the shadows for each cluster with a soft shadow algorithm. Due to an adjustable initial cluster distance, the performance and quality can be trade off. The results showed, that LightCluster can be used to increase the rendering performance while maintaining an acceptable shadow quality in many cases.

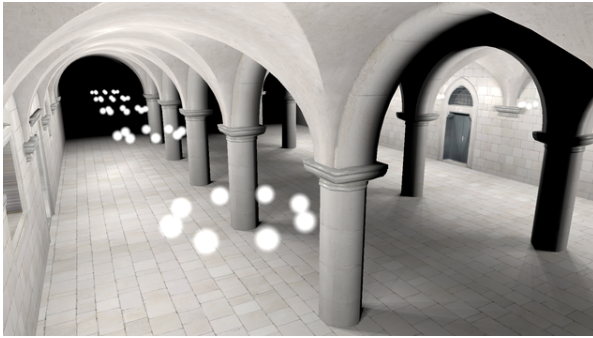
In future work, we wish to extend our implementation with different light types, such as spot lights. This will result in a different clustering metric as new properties of the light sources must be incorporated, such as the direction vector. Furthermore, we wish to try other shadow techniques for omnidirectional light sources, such as Paraboloid Shadow Mapping [Bra02a] and Tetrahedron Shadow Mapping [Hun10a].

## 7 ACKNOWLEDGMENTS

The Dabrovic Sponza and Cornell Box scene is downloaded from [McG13a]. The Restaurant scene is based on the restaurant model from [IDS13a]. The work of A. Klein is funded by MBDA Germany.

## 8 REFERENCES

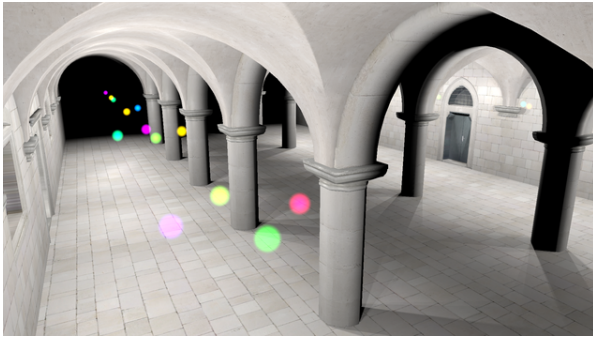
- [Bra02a] Brabec S., Annen T., Seidel H.-P. Shadow Mapping for Hemispherical and Omnidirectional Light Sources. In *Proceedings Computer Graphics International*, Springer, 397-408, 2002.
- [Dac05a] Dachsbacher C., Stamminger M. Reflective Shadow Maps. In *Conf.proc. I3D 2005*, 203-231, 2005.
- [Don09a] Dong Z., Grosch T., Ritschel T., Kautz J., Seidel H.-P. Real-time indirect illumination with clustered visibility. In *Conf.proc. VMV 2009*, 187-196, 2009.
- [Eis11a] Eisemann E., Schwarz M., Assarsson U., Wimmer M. *Real-Time Shadows*, Taylor & Francis, 2011.
- [Fer05a] Fernando R. Percentage-Closer Soft Shadows. *ACM SIGGRAPH 2005 Sketches*, 2005.
- [Ger04a] Gerasimov P. S. *Omnidirectional Shadow Mapping*, GPU Gems, Addison-Wesley, 193-203, 2004.
- [Gre93a] Greene N., Kass M., Miller G. Hierarchical Z-buffer visibility. In *Conf.proc. SIGGRAPH '93*. ACM, 231-238, 1993.
- [Haš07a] Hašan M., Pellacini F., Bala K., Matrix row-column sampling for the many-light problem. In *Conf.proc. SIGGRAPH '07*, ACM, 2007.
- [Haš08a] Hašan M., Velázquez-Armendáriz E., Pellacini F., Bala K., Tensor Clustering for Rendering Many-Light Animations. In *Conf.proc. EGSR '08*, ACM, 1105-1114, 2008.
- [Hun10a] Hung-Chien L., Shadow Mapping for Omnidirectional Light Using Tetrahedron Mapping. *GPU Pro*, A K Peters, 455-475, 2010.
- [IDS13a] IDST Render, accessed March 27, 2013. <http://idst-render.com/scenes.html>.
- [Kel97a] Keller A. Instant Radiosity. In *Conf.proc. SIGGRAPH '97*, ACM, 49-56, 1997.
- [Lai07a] Laine S., Lehtinen J., Kontkanen J. Incremental Instant Radiosity for real-time indirect illumination. In *Conf.proc. EGSR '07*, ACM, 277-286, 2007.
- [Lau10a] Lauritzen A. Deferred Rendering for Current and Future Rendering Pipelines. *Beyond Programmable Shading*, SIGGRAPH 2010, 2010.
- [McQ67a] MacQueen J. B. Some methods for classification and analysis of multivariate observations. *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 281-297, 1967.
- [McG13a] McGuire M., Computer Graphics Archive, accessed March 27, 2013. <http://graphics.cs.williams.edu/data>.
- [Ols11a] Olsson O., Assarsson U. Tiled Shading. *Journal of Graphics, GPU, and Game Tools*, 235-251, 2011.
- [Ree87a] Reeves W. T., Salesin D. H., Cook R. L. Rendering antialiased shadows with depth maps. In *Conf.proc. SIGGRAPH '87*, ACM, 283-291, 1987.
- [Rit08a] Ritschel T., Grosch T., Kim M., Seidel H.-P., Dachsberger C., Kautz J. Imperfect shadow maps for efficient computation of indirect illumination. In *Conf.proc. SIGGRAPH Asia '08*, ACM, 2008.
- [Wal05a] Walter B., Fernandez S., Arbee A., Bala K., Donikian M., Greenberg D. P. Lightcuts: a scalable approach to illumination. In *Conf.proc. SIGGRAPH '05*, ACM, 1098-1107, 2005.
- [Wil78a] Williams L. Casting curved shadows on curved surfaces. In *Conf.proc. SIGGRAPH '78*, ACM, 270-274, 1978.
- [Wol57a] Wolfowitz J. The Minimum Distance Method, *Annals of Mathematical Statistics*, vol. 28, 75-88, 1957.



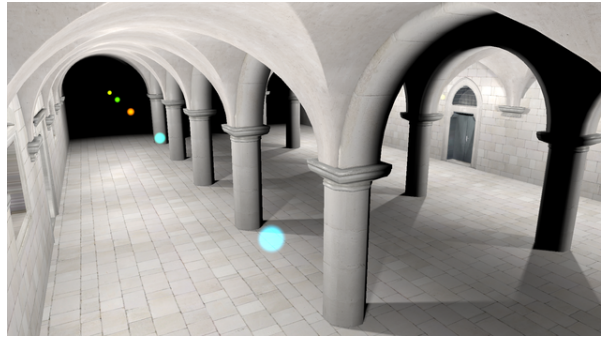
(a) The light distribution of 80 lights with  $L_{att} = 7$  each.



(b) The resulting reference solution at 97.8 ms.



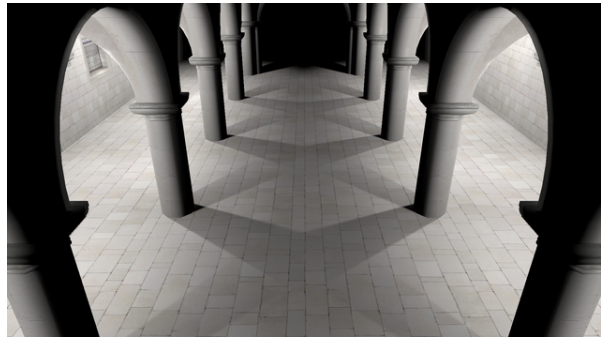
(c) The clustering for  $d_{init} = 0.5$  with 26 clusters.



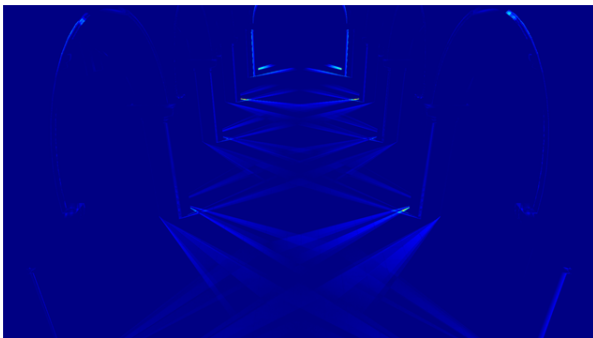
(d) The clustering for  $d_{init} = 1.6$  with ten clusters.



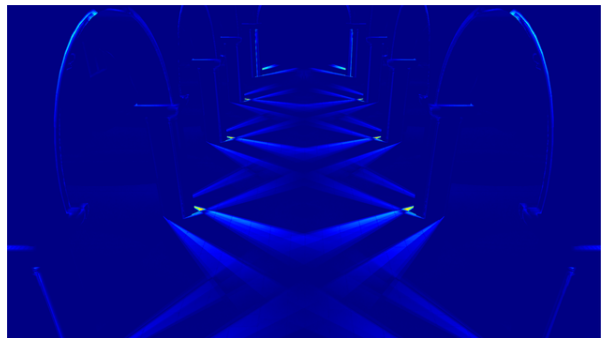
(e) The resulting shadows for the cluster distribution at 43.1 ms.



(f) The resulting shadows for the cluster distribution at 20.7 ms.

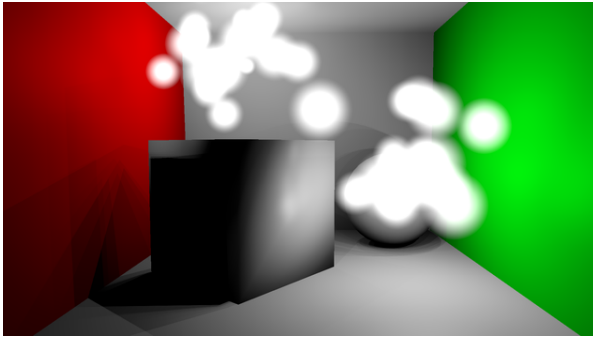


(g) The error for 26 clusters compared to the reference solution.

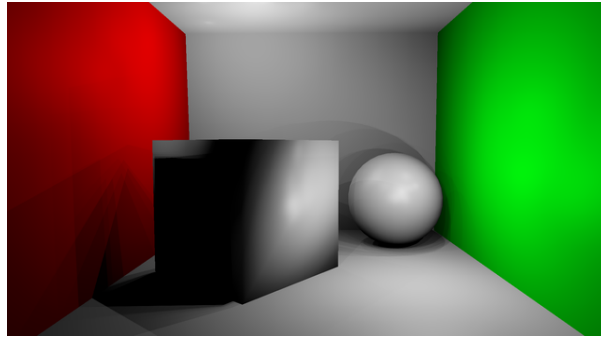


(h) The error for ten clusters compared to the reference solution.

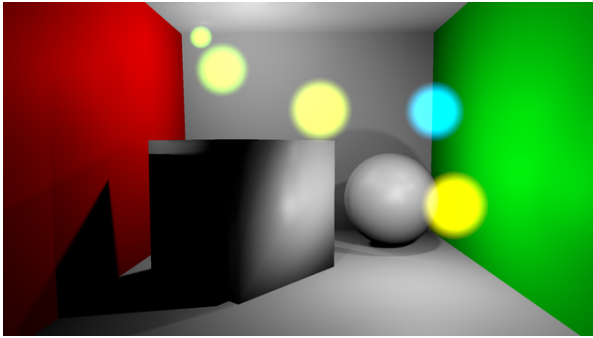
Figure 7: Comparison for the Dabrovic Sponza scene.



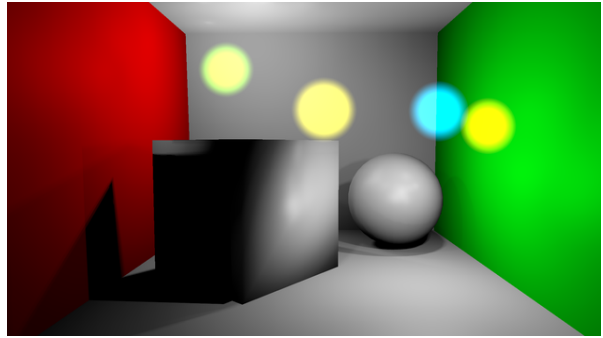
(a) The light distribution of 32 lights with random  $L_{att}$ .



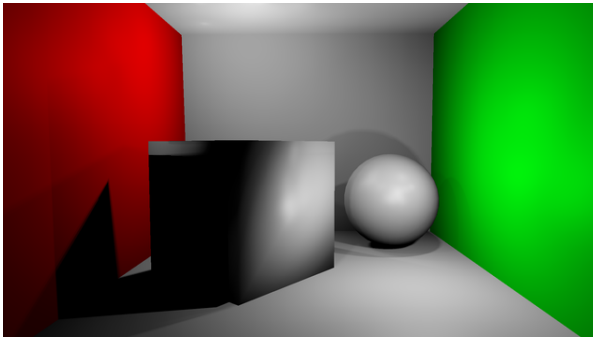
(b) The resulting reference solution at 20.0 ms.



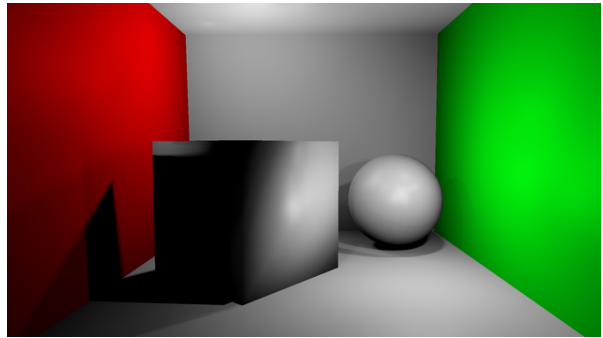
(c) The clustering for  $d_{init} = 0.5$  with five clusters.



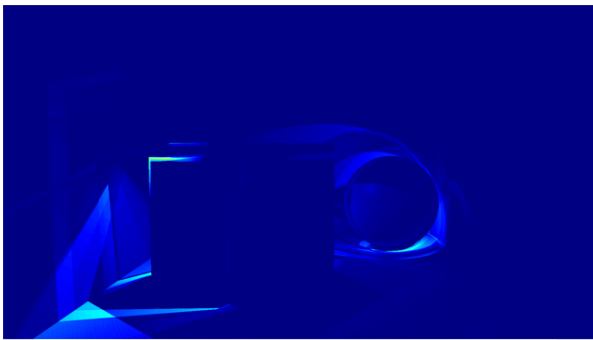
(d) The clustering for  $d_{init} = 0.6$  with four clusters.



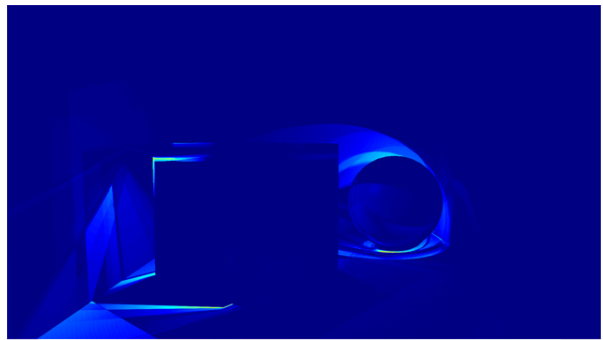
(e) The resulting shadows for the cluster distribution at 12.1 ms.



(f) The resulting shadows for the cluster distribution at 11.1 ms.



(g) The error for five clusters compared to the reference solution.



(h) The error for four clusters compared to the reference solution.

Figure 8: Comparison for the Cornell Box scene.





(a) The light distribution of fourteen lights with  $L_{att} = 2$  each.



(b) The resulting reference solution at 39.3 ms.



(c) The clustering for  $d_{init} = 0.3$  with nine clusters.



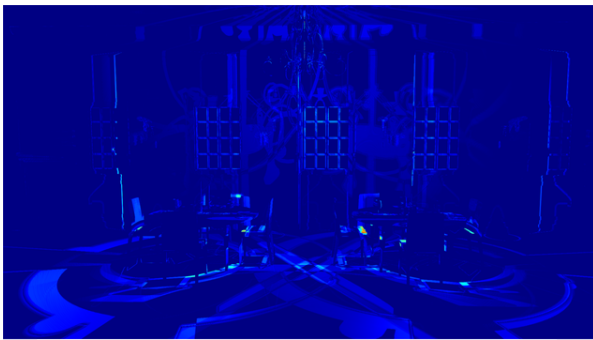
(d) The clustering for  $d_{init} = 0.4$  with eight clusters.



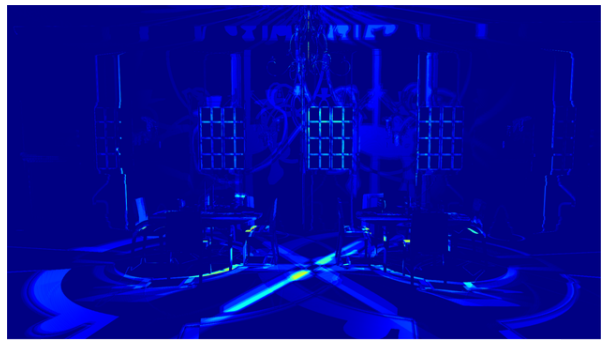
(e) The resulting shadows for the cluster distribution at 32.9 ms.



(f) The resulting shadows for the cluster distribution at 30.5 ms.



(g) The error for nine clusters compared to the reference solution.



(h) The error for eight clusters compared to the reference solution.

Figure 9: Comparison for the Restaurant scene.

# Using holistic features for scene classification by combining classifiers

Kelly Assis de Souza Gazolli

Universidade Federal do Espírito Santo  
Brazil, Vitória, ES  
kasouza@ifes.edu.br

Evandro Ottoni Teattini Salles

Universidade Federal do Espírito Santo  
Brazil, Vitória, ES  
evandro@ele.ufes.br

## ABSTRACT

Scene classification is a useful, yet challenging problem in computer vision. Two important tasks for scene classification are the image representation and the choice of the classifier used for decision making. This paper proposes a new technique for scene classification using combined classifiers method. We run two classifiers based on different features: GistCMCT and spatial MCT and combine the output results to obtain the final class. In this way, we improve accuracy, by taking advantage from the qualities of the two descriptors, without increasing the final size of the feature vector. Experimental results on four used datasets demonstrate that the proposed methods could achieve competitive performance against previous methods.

## Keywords

visual descriptor, gist, CMCT, CENTRIST, combining classifiers.

## 1. INTRODUCTION

The scene classification is an important topic in computer vision. However, while classifying a scene is not a problem for humans, it is quite a challenging task for computers. Among the reasons is the significant intra-class variations, since a scene is composed of several entities often organized in an unpredictable layout. Moreover, there are other obstacles such as, variations in lighting and scale, different view angles, occlusion and dynamic backgrounds. All these factors make it difficult to find a unique representation for a scene category that encompasses all possible variations for scenes belonging to it. The holistic approach is a common method for scene classification. This approach does not require explicit segmentation of image and objects, the image is considered as a whole. Oliva and Torralba [Oli05a][Oli01a] showed that scenes which belong to the same category, normally, have the same spatial layout properties (naturalness, openness, expansion, depth, roughness, complexity,

ruggedness, symmetry) and proposed a holistic approach to build the "gist" of the scene. Wu and Rehg [Wu11a] also used a holistic approach and proposed CENTRIST (Census Transform Histogram), a representation that captures structural properties, such as, rough geometry and generalizability, by modeling distribution of local structures. In this sense, a modification of CENTRIST was proposed, the CMCT (Contextual Mean Census Transform) [Gaz12a]. In this representation the modeling of distribution of local structures is combined with contextual information.

Another approach used in scene classification is the spatial pyramid representation [Lab06a] that captures useful information, like regularities in the image and spatial arrangement of the features, in order to improve the classification task. Wu and Rehg, in [Wu11a], proposed the spatial PACT (Principal component Analysis of Census Transform histograms) a technique which combines spatial information with CENTRIST and improve the classification performance.

In this paper, we work with two different features: GistCMCT [Gaz13a] and spatial MCT. GistCMCT combines the vector generated from gist [Oli01a] [Oli06a] with the vector generated from CMCT [Gaz12a] in a new vector with the aim of improve classification results as much as for indoor as for outdoor scenes. In the other hand, spatial MCT uses

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

spatial information, which also improves classification performance. With the purpose of take advantages of the qualities of both type of features without increasing the size of features vectors, we used multiple classifiers and combine their results to find the most likely class.

The rest of this article is structured as follows: Section II presents the theoretical background, Section III presents the proposed technique, Section IV presents the experimental results and, finally, in Section V the conclusion is provided.

## 2. THEORETICAL BACKGROUND

### 2.1 Modified Census Transform

The Modified Census transform (MCT) [Fro04a] is inspired on Census Transform [Zab94a], a nonparametric local transform originally designed for computing visual correspondence, and it was proposed by Fröba and Ernst with the aim of overcome some weakness of Census Transform. First, The Modified Census Transform,  $I(x)$ , computes a mean  $\bar{I}(x)$  over  $3 \times 3$  window of pixels. So, every pixel in the  $3 \times 3$  window is then compared with  $\bar{I}(x)$ . If the pixel is bigger than or equal to  $\bar{I}(x)$ , a bit 1 is set in the corresponding location, otherwise, a bit 0 is set, as follows

$$I(x) = \otimes_{y \in N'(x)} \zeta(I(y), \bar{I}(x)),$$

$$\zeta(m, n) = 1, m \geq n; \zeta(m, n) = 0, \text{otherwise}$$

where  $\otimes$  represents concatenation operation,  $\bar{I}(x)$  is the mean of the intensity values in the  $3 \times 3$  window of pixels centered at  $x$ ,  $I(y)$  is the gray value of the pixel at  $y$  position and  $N'(x)$  is a local spatial neighborhood of the pixel at  $x$  so that  $N(x) = N'(x) \cup x$ . In the Modified Census Transform technique, 9 bits are generated and converted to a decimal number in  $[0, 511]$ , namely, here, MCT.

### 2.2 CMCT - Contextual Mean Census Transform

Contextual information provides a support for scene classification. A white image region is likely to be the cloud if it is in a sky area while could be snow if it is next to a mountain. With the aim of adding contextual information to MCT descriptor, Gazolli and Salles [Gaz12a] proposed the Contextual Mean Census Transform (CMCT), which integrates contextual information with local structures information for differentiating windows in the image that have similar structures, but have significant difference in their neighborhood. For accomplishing this task, this approach considers information from

neighborhood windows in the MCT computation, by creating a new local structure from the local structure of the window and from the local structures of its neighboring windows. The information from the outside of the window is called context.

The MCT used in the CMCT differs slightly from the original, because  $\bar{I}(x)$  is not compared with the center pixel. Thus, MCT generates 8 bits, instead of 9, which are converted to a decimal number in  $[0, 255]$ . In order to differentiate Modified Census Transform with 9 bits from Modified Census Transform with 8 bits this last is referred as MCT8.

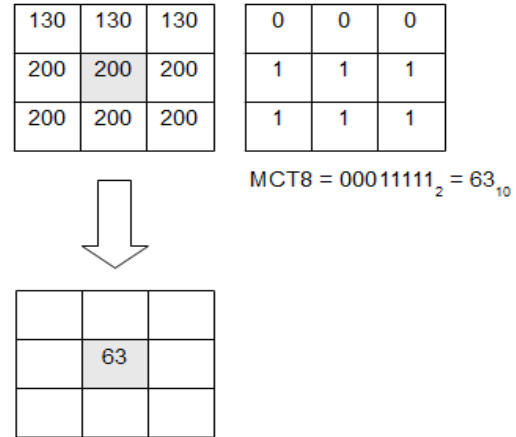


Figure 1. Pixel value is replaced by the MCT8 for obtaining contextual information.

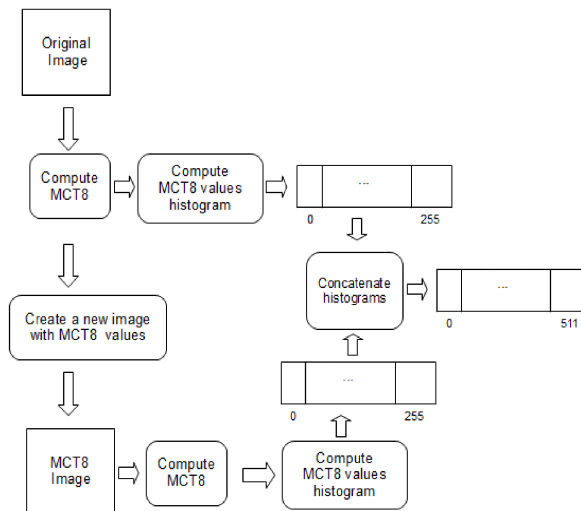
The steps for the Contextual Mean Census Transform (CMCT) generation are as follow. First, MCT8 is computed for all pixels. Then, a histogram of MCT8 is obtained. A new image is created in which the original image pixels are replaced by the correspondent MCT8 values as shown in Fig. 1.

In the sequel, the MCT8 is computed on the new image pixels and a new histogram is generated. Then, the MCT8 histogram for the original image histogram and the the MCT8 histogram for the new image are concatenated, generating a new descriptor. The whole process is schematized in Fig. 2.

### 2.3 GistCMCT

Under the assumption that is not necessary identify the objects that make up a scene to identify the scene, Oliva and Torralba [Oli01a] proposed a holistic approach to build the 'gist' of the scene using a low-dimensional representation of perceptual dimensions, a set of global image properties, such as naturalness, openness, roughness, expansion and ruggedness. This approach uses those perceptual dimensions to define the functionality of the scene location in the three-dimensional space. The set of perceptual dimensions is named Spatial Envelope. As gist is a holistic representation of the scene structure, it does not require explicit segmentation of image and objects.





**Figure 2. Contextual descriptor extraction process.**

Gist represents the shape of the scene by computing stable spatial structures within images that reflect functionality of the location. Besides that, gist has a certain weakness in recognizing indoor scenes, but is quite efficient in the recognition of outdoor scenes [Wu11a]. In the other hand, CMCT summarizes local shape information. This descriptor represents structural properties through the distribution of local structures (for example, the amount of local structures that are local horizontal edge) [Wu11a] which helps in the classification of man-made environments, including, indoor environments.

Aiming the improvement of classification results as much as for indoor as for outdoor scenes, the GistCMCT [Gaz13a] was proposed. The GistCMCT combines the vector generated by gist [Oli01a] [Oli06a] with the vector generated by CMCT [Gaz12a] in a new one vector. The new vector gathers qualities from gist and CMCT, reaching, hence, a better performance in classifying scenes.

The information about the perceptual dimensions can be extracted from linear filters [Oli05a]. In this work, the gist descriptor was obtained from a bank of Gabor filters. We used 4 scales and 8 orientations and the outputs of the filters were downsampled in a 4 x 4 grid, generating a vector with 512 positions. As CMCT also generates a 512 size vector, the final descriptor is a vector with 1,024 positions.

## 2.4 Spatial Information

Lazebnik *et al.* showed in [Lab06a] that the spatial arrangement of the features and the regularities in image composition provides powerful cues for scene classification tasks. In order to incorporate spatial information, Wu and Rehg [Wu11a] proposed a spatial representation which is based on the Spatial Pyramid Matching scheme in [Lab06a]. In this

representation, the image is divided in blocks and the correspondent results in these blocks are concatenated. However, in order to avoid artifacts created by the non-overlapping division, the blocks division is shifted. Besides that, the image is resized between different levels so that all blocks contain the same number of pixels.

In this work, we adopted the representation proposed in [Wu11a]. However, instead of using CENTRIST, we apply MCT [Fro04a], once by using MCT it is possible differentiating structures that are considered equal by CENTRIST. The MCT vectors obtained in all blocks are then concatenated to form an overall feature vector. After obtaining the final vector, we used Principal Component Analysis to reduce its dimensionality, in the same way as [Wu11a]. We use 3 levels of spatial information, as [Wu11a], which generates 31 blocks and reduce the dimensionality of each block from 512 to 40. We also adopted the extra information (mean and standard deviation of pixel blocks). We refer the spatial pyramid of MCT as spatial MCT.

## 3. ADDING SPATIAL INFORMATION BY COMBINING CLASSIFIERS

As we presented in Section 2.3, GistCMCT combines two different feature descriptors in order to improve both indoor and outdoor scene classification. However, GistCMCT does not consider the spatial layout of the features in image. Spatial MCT, on the other hand, does, and, as discussed in Section 2.4, this type of information can help improving the classification results.

According to [Ho94a], the classification accuracy could be improved by using features and classifiers of

different types simultaneously, through multiple classifiers. It has been observed that features and classifiers of different types complement one another in classification performance, i.e., the sets of patterns misclassified by different classifiers would not necessarily overlap [Kit98a].

With the aim of improve the performance of scene classification by combining the qualities of GistCMCT and spatial MCT without increasing the size of the feature vector, we adopted the combining classifiers strategy. In this way, each of these feature sets trains an individual classifier and the results of these classifiers are used for decision making by combining their individual opinions to derive a consensus decision.

The classifier adopted for both descriptors is the SVM (Support Vector Machine), a pattern classifier introduced by Vapnik [Vap98a], with Histogram Intersection kernel (HIK) [Wu09a]. Despite of using the same type of classifier, the features extracted from the images are unique to each one, since each classifier uses its own representation of the image (GistCMCT or spatial MCT). In this way, we integrate physically different types of features.

The implementation of a multiple classifier system implies the definition of a rule (combining rule) for determining the most likely class, on the basis of the class attributed by each single classifier [Fog07a]. For combining the individual opinion from each classifier, we used the combination rules Max, Median and Product presented in [Kit98a]. For simplicity, we assume that the results from each classifier are statistically independent. For equiprobable distribution classes, the selected  $\omega$  class is the one that satisfies to the following equations [Kit98a]

- Max:

$$\max_{i=1}^R P(\omega_j|x_i) = \max_{k=1}^m \max_{i=1}^R P(\omega_k|x_i)$$

- Median

$$\text{med}_{i=1}^R P(\omega_j|x_i) = \max_{k=1}^m \text{med}_{i=1}^R P(\omega_k|x_i)$$

- Product:

$$\prod_{i=1}^R P(\omega_j|x_i) = \max_{k=1}^m \prod_{i=1}^R P(\omega_k|x_i)$$

where  $m$  is the number of possibles classes  $(\omega_1, \dots, \omega_m)$ ,  $x_i$  is the measurement vector used by the  $i$ th classifier, and  $R$  is the number of classifiers, in our case,  $R=2$ .

## 4. EXPERIMENTS

In this section, we investigate the effectiveness of our representations and compare them with existing works.

### 4.1 Datasets and Setup

Our descriptor has been tested on four data sets. These datasets are described bellow:

- 8-category scenes provided by Oliva and Torralba [Oli01a]. This dataset contains 2,688 color images, divide into 8 categories; with the number of images in each category ranging from 260 to 410. The 8 categories are: coast (360 images), forest (328 images), mountain (274 images), open country (410 images), highway (260 images), inside city (308 images), tall building (356 images) and street (292 images). The size of each image is 256 x 256.

- 15-category dataset [Lab06a], which is an extension of dataset described above by adding 7 new scene categories: bedroom (216 images), kitchen (210 images), living room (289 images), office (215 images), suburb (241 images), industrial (311 images) and store (315 images). This dataset contains 4,486 gray-values images in total. The image size is approximately 300 x 250. Fig. 3 depicts the samples from this dataset

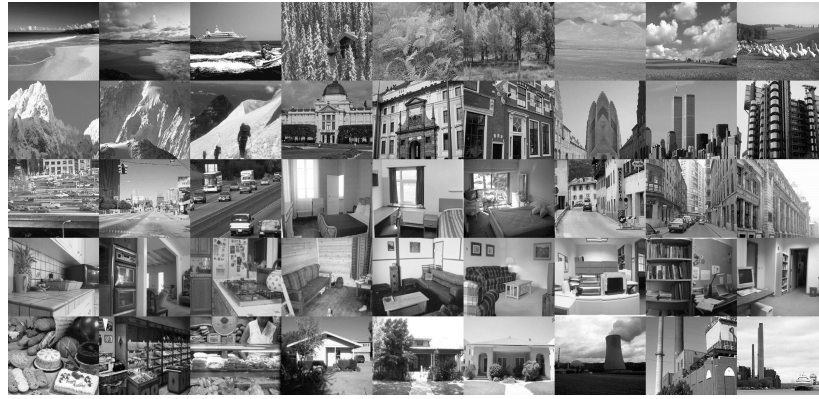
- 8-class sports event [Li07a]. This dataset contains 1,579 images of eight sports: badminton, bocce, croquet, polo, rock climbing, rowing, sailing and snowboarding. The number of images in each category varies from 137 to 250. Fig. 4 depicts the samples from this dataset.

- 67-class indoor scene recognition [Qua09a]. This dataset contains 15,620 images. The scenes varies from corridor to bakery. This dataset poses a challenging classification problem [Qua09a].



**Figure 4. Two images from each 8 sports events categories. The categories are: badminton, bocce, croquet, polo, rock climbing, rowing, sailing and snowboarding (from top to bottom and left to right).**

In the experiment, each dataset category is split randomly into a training set and a test set. The random splitting is repeated 5 times, and the average accuracy is reported, as adopted by [Wu11a]. All color images were converted to gray scale. For



**Figure 3. Three images from each 15 scene categories. The categories are: coast, forest, open country, mountain, inside city, tall building, highway, bedroom, street, kitchen, living room, office, store, suburb and industrial (from top to bottom and left to right).**

training and classification, we adopted SVM (Support Vector Machine) and used the libSVM [Chal1a] package modified by [Wu09a], which offers the option of choose the estimated multi-class probability as output [Wu04a].

Besides that in all experiments performed, we employed Histogram Intersection kernel (HIK) [Wu09a] Support Vector Machine, because the best results were achieved when using this kernel type. For testing gist we used the Lear's gist implementation<sup>1</sup>.

## 4.2 Results on 15-category Dataset

In this dataset an amount of 100 images in each category are used for training and the remaining images constitute the testing set, as in previous researches. When using our approach with product rule, we achieve  $86.25 \pm 0.51\%$  accuracy in this dataset. Fig. 5 presents the confusion matrix from one run on 15-class scene dataset. We observe that the highest recognition rate was achieved for suburb class. The biggest confusion happens between bedroom and living room, which have similar elements. Humans may confuse them due to the small inter-class variation.

Table 1 presents the classification performance of the proposed method on 15-category dataset compared with existing methods in literature. All approaches in this section used the SVM classifier.

In [Lab06a], it is proposed the spatial pyramid method (SPM), a extension of an orderless bag-of-features image representation. The best performance of SPM is achieved with vocabulary size = 400 and level number = 3 (with leads to a 34000 dimensions final vector). Ergul and Arica [Erg10a] proposed a scene classification method which combines two popular approaches in the literature: Spatial Pyramid

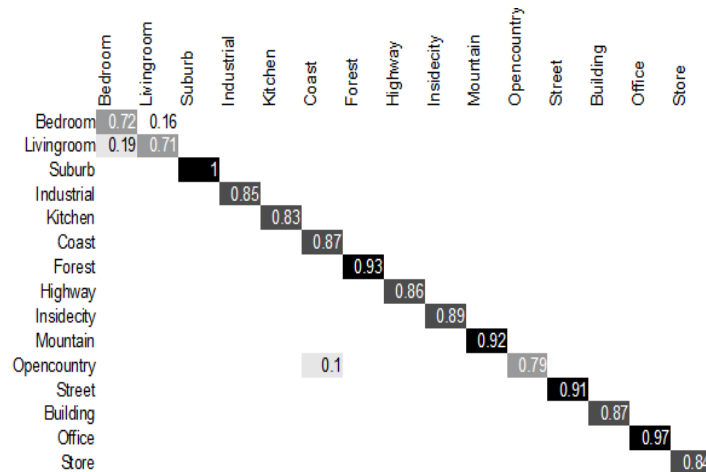
Matching (SPM) and probabilistic Latent Semantic Analysis (pLSA) modeling, the Cascaded pLSA. The results presented here refers to a 5900 dimensions vector.

Method	Accuracy(%)
SPM [Lab06a]	$81.4 \pm 0.5$
Cascaded pLSA [Erg10a]	83.31
CENTRIST (3 levels) [Wu11a]	$83.88 \pm 0.76$
LDBP [Men12a]	$84.10 \pm 0.96$
CBoW_SL [Li11a]	85.1
LBP + Semantic [Li11b]	85.1
GistCMCT [Gaz13a]	$82.72 \pm 0.55$
Spatial MCT	$84.47 \pm 0.63$
<b>Ours (rule: max)</b>	<b><math>86.02 \pm 0.46</math></b>
<b>Ours (rule: median)</b>	<b><math>86.30 \pm 0.62</math></b>
<b>Ours (rule: product)</b>	<b><math>86.25 \pm 0.51</math></b>

**Table 1. Comparison classification results for 15 categories with existent works.**

In [Wu11a] the CENTRIST with 3 levels, 40 eigenvectors and 1302 dimensions reached the best result. In [Men12a], the histogram of local transform represents a scene images. This histogram is an extended version of census transform histogram, by applying Local Difference Binary Pattern (LDBP). Besides that, this approach also uses a spatial pyramid representation. In this approach, the final descriptor has 840 dimensions. In [Li11a] a novel contextual Bag-of-Words (CBoW) representation was proposed to model two kinds of typical contextual relations between local patches: a semantic conceptual relation and a spatial neighboring relation. The best performance is achieved when the proposed CBoW is combined with the spatial layout (CboW\_SL), with leads to a 2250 dimensions vector. Li and Dewen [Li11b] proposed a

<sup>1</sup> Available in <http://lear.inrialpes.fr/software>



**Figure 5. Confusion matrix from one run for 15-class scene recognition experiment. Only rates higher or equal than 0.1 are shown in the figure.**

scene classification approach based on combining low-level, by using Local Binary Pattern (LBP), and semantic modeling strategies, local feature extraction and codebook generation. The codebook size was not informed.

As one can see, the proposed approach reached better results than the above methods, including GistCMCT and Spatial MCT separately. With respect to the three combining rules here used, the results were close, despite the max rule achieved the worst result.

### 4.3 Results on 8-category Dataset

In this dataset an amount of 100 images in each category are used for training and the remaining images constitute the testing set, as in previous researches. In the 8-category scene class our method, with product rule, achieves  $88.95 \pm 0.49\%$  accuracy. Table 2 shows experimental results for 8-category dataset. As one can see, the proposed method overcomes all aforementioned methods.

Method	Accuracy (%)
Gist [Oli01a]	$82.60 \pm 0.86$
Novel Gist [Men10a]	$86.60 \pm 0.53$
CENTRIST (3 levels) [Wu11a]	$86.20 \pm 1.02$
GistCMCT [Gaz13a]	$85.82 \pm 0.93$
Spatial MCT	$87.65 \pm 0.24$
<b>Ours (rule: max)</b>	<b><math>88.51 \pm 0.33</math></b>
<b>Ours (rule: median)</b>	<b><math>88.83 \pm 0.46</math></b>
<b>Ours (rule: product)</b>	<b><math>88.95 \pm 0.49</math></b>

**Table 2. Experimental results for 8 scene categories dataset.**

The Novel Gist [Men10a] is an extension of census transform and also uses spatial information. In this technique the histograms of upper pattern and lower pattern are computed and then concatenated. The

experiments reported uses SMV classifier and a 1610 dimensions vector.

Once again, the proposed approach reached better results than GistCMCT and Spatial MCT separately. With respect to combining rule, the results were close, but, as in the 15 scene datasets, the max rule reached the worst result.

### 4.4 Results on 8-class Sports Event

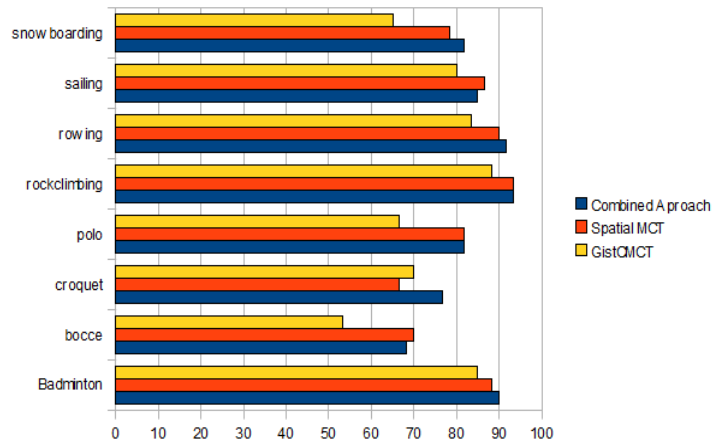
Following [Li07a], in this dataset, we use 70 images per class for training and 60 for testing. Table 3 shows the results for this dataset.

Method	Accuracy(%)
Scene + object [Li07a]	73.4
CENTRIST (3 levels) [Wu11a]	$78.25 \pm 1.27$
GistCMCT [Gaz13a]	$74.37 \pm 1.34$
Spatial MCT	$80.63 \pm 1.17$
<b>Ours (rule: max)</b>	<b><math>80.96 \pm 0.86</math></b>
<b>Ours (rule: median)</b>	<b><math>81.33 \pm 1.19</math></b>
<b>Ours (rule: product)</b>	<b><math>81.54 \pm 1.62</math></b>

**Table 3. Experimental results for 8 class sports event dataset.**

In [Li07a], in which the event classification is a result of scene environment classification, object categorization, a manual segmentation and finally, the object labels are used as additional inputs.

Our classification methods achieves  $81.54 \pm 1.62\%$  and, so, for this dataset, the difference between the results for the proposed approach and for Spatial MCT is very small, less than 1%. Fig. 6 shows the comparison results among GistCMCT, spatial MCT and the proposed approach for one run experiment. We observe that the recognition rate, in the proposed approach, for the classes Sailing and Bocce are worst



**Figure 6. Classification rates by class for GistCMCT, spatial MCT and the combined approach from one run scene recognition experiment in the 8-class sports event dataset.**

than spatial MCT. For these cases the information brought from GistCMCT doesn't help at all.

#### 4.5 Results on 67-class Indoor Scene Recognition

In this dataset, we use 80 images in each category for training and 20 images for testing following [Qua09a]. All approaches presented here use the SVM classifier. Table 4 presents the results for this dataset.

Method	Accuracy(%)
Gist [Oli01a]	21
Global + local [Qua09a]	25
CENTRIST (3 levels) [Wu11a]	$36.88 \pm 1.10$
Hibrid Representation [Niu10a]	40.19
GistCMCT [Gaz13a]	$33.60 \pm 1.30$
Spatial MCT	$38.58 \pm 1.44$
<b>Ours (rule: max)</b>	<b><math>40.45 \pm 1.41</math></b>
<b>Ours (rule: median)</b>	<b><math>41.83 \pm 1.22</math></b>
<b>Ours (rule: product)</b>	<b><math>42.42 \pm 1.32</math></b>

**Table 4. Experimental results for 67 class indoor scene categories dataset.**

The experiments performed by [Qua09a] used local and global information to represent the scenes and the feature dimensions depends on the number of the Regions of Interest. In [Niu10a], a hybrid image representation by combining the global information with the local structure of the scene was proposed, generating a 34692 dimensions vector.

By using the proposed method we achieve  $42.42 \pm 1.32\%$  employing the product rule and, as one can see, for this dataset, which contains only indoor scenes, the difference between the results for the

proposed approach and for Spatial MCT is close to 4%.

With respect to combining rule, the max rule achieved the worst results again.

#### 5. CONCLUSION

In this paper, we proposed a new technique for scene classification by combining two classifiers based on different features, GistCMCT and spatial MCT, to improve classification performance. Combining classifiers allows the union of the complementary qualities of the two image descriptors without increasing the size of the feature vector. The experiments presented show the potential applicability of the technique. Nevertheless, when dealing with the recognition of events, the proposed approach did not bring a great vantage, as one can see in the results of events sports dataset, in which the results reached were very close to the spatial MCT approach results.

Besides the performance improvement, GistCMCT and spatial MCT are a holistic and low-dimensional representation of the structure of a scene and, also, don't require quantization of the data, as in the bag-of-features approach, which could be a computational expensive process.

In addition, the proposed approach is flexible and enables the use of different SVM kernels for each descriptor or, even, different kinds of classifiers, which can help to reach performance improvement through the choice of the most appropriate classifiers for each descriptor.

In future work, we plan to obtain contextual information from the different levels in the spatial layout and use other ways for combining classifiers by employing machine learning.

## 6. ACKNOWLEDGMENTS

Kelly Gazolli gratefully acknowledge the support from IFES - Instituto Federal do Espírito Santo.

## 7. REFERENCES

- [Bos06a] Bosch, A. Zisserman, A. and Muñoz, X. Scene classification via plda. In Proceedings of the 9th European conference on Computer Vision, ECCV'06, pages 517–530, 2006.
- [Cha11a] Chang, C.-C. and Lin, C.-J. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, pages 1–27, 2011.
- [Erg10a] Ergul, E. and Arica, N. Scene classification using spatial pyramid of latent topics. In Proceedings of the 2010 20<sup>th</sup> International Conference on Pattern Recognition, ICPR'10, pages: 3603–3606, 2010.
- [Fog07a] Foggia, P., Percannella, G., Sansone, C. and Vento, M. Evaluating classification reliability for combining classifiers. In Proceedings of the 14th International Conference on Image Analysis and Processing, ICIAP '07, pages 711–716, 2007.
- [Fro04a] Fröba, B. and Ernst, A. Face detection with the modified census transform. In Proceedings of the Sixth IEEE international conference on Automatic face and gesture recognition, pages 91–96, 2004.
- [Gaz12a] Gazolli, K. and Salles, E. A contextual image descriptor for scene classification. In Online Proceedings on Trends in Innovative Computing, pages 66–71, 2012. [Online]. Available: <http://www.mirlabs.net/ict12/download/Paper13.pdf>
- [Gaz13a] Gazolli, K. and Salles E. Combining holistic descriptors for scene classification. In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP 2013), pages 315–320, 2013.
- [Ho94a] Ho, K., Hull, J. J. and Srihari, S. N. Decision combination in multiple classifier systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 66–75, 1994.
- [Kit98a] Kittler, J., Hatef, M., Duin, R. P. W. and Matas, J. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 226–239, 1998.
- [Lab06a] Lazebnik, S., Schmid, C. and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '06, pages 2169–2178, 2006.
- [Li07a] Li, L.-J. and Fei-Fei, L. What, where and who? Classifying events by scene and object recognition. In IEEE 11th International Conference on Computer Vision, pages 1–8, 2007.
- [Li11a] Li, T., Mei, T., Kweon, I.-S. and Hua, X.-S. Contextual bag-of-words for visual categorization. *Circuits and Systems for Video Technology, IEEE Transactions on*, pages 381–392, 2011.
- [Li11b] Li, Z. and Dewen, H. Scene classification combining low-level and semantic modeling strategies. In *Digital Manufacturing and Automation (ICDMA)*, 2011 Second International Conference on, pages 1071–1075, 2011.
- [Men10a] Meng, X. and Wang, Z.. Rapid scene categorization using novel gist model. In *Information Engineering and Computer Science (ICIECS)*, 2010 2nd International Conference on, 2010.
- [Men12a] Meng, X., Wang, Z. and Wu, L. Building global image features for scene recognition. *Pattern Recogn.*, pages 373–380, 2012.
- [Niu10a] Niu, Z., Zhou, Y. and Shi, K.. A hybrid image representation for indoor scene classification. In *Image and Vision Computing New Zealand (IVCNZ)*, 2010 25th International Conference of, pages 1–7, 2010.
- [Oli05a] Oliva, A. . Gist of the scene. *Nature*, pages 251–257, 2005.
- [Oli01a] Oliva A. and Torralba, A.. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, pages: 145–175, 2001.
- [Oli06a] Oliva, A. and Torralba, A.. Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, page 23–36, 2006.
- [Qua09a] Quattoni, A. and Torralba, A. Recognizing indoor scenes. In *Proceedings IEEE CS Conf. Computer Vision and Pattern Recognition*, pages 413–420, 2009.
- [Vap98a] Vapnik, V. The support vectormethod of function estimation. *Nonlinear Modeling advanced blackbox techniques* Suykens JAK Vandewalle J Eds, pages 55–85, 1998.
- [Wu09a] Wu, J. and Rehg, J.M. Beyond the euclidean distance : Creating effective visual codebooks using the histogram intersection kernel. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 630–637, 2009.
- [Wu11a] Wu, J. and Rehg, J. M. Centrist: A visual descriptor for scene categorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1489–1501, 2011.
- [Wu04a] Wu, T.-F., Lin, C.-J. and Weng, R. C. Probability Estimates for Multi-class Classification by Pairwise Coupling. *Journal of Machine Learning Research* 5, pages 975-1005, 2004.
- [Zab94a] Zabih, R. and Woodfill, J. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the third European conference on Computer Vision - Volume 2, ECCV '94*, pages 151–158, 1994.



# Frequency-based Progressive Rendering of Continuous Scatterplots

Vladimir Molchanov

v.molchanov@jacobs-  
university.de

Alexey Fofonov

a.fofonov@jacobs-niversity.de

Lars Linsen

l.linsen@jacobs-university.de

Jacobs University, Campus Ring 1, 28759 Bremen, Germany

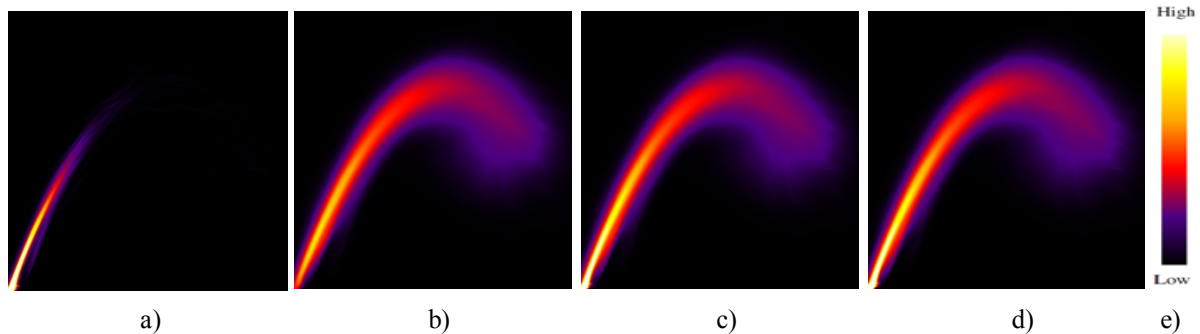


Figure 1: Hybrid approach for continuous scatterplot computation (“Bucky Ball” dataset): Small splats are rendered directly (a), large splats are computed via their spectral representation (b), and both textures are added to produce the final result (c). Result is compared to direct rendering of all splats (d). Color scheme is shown (e).

## ABSTRACT

Continuous scatterplots are a consistent tool for the visual representation and exploration of continuous multivariate data defined on a continuous domain. Due to the complexity of the construction algorithm, application of continuous scatterplots is limited in terms of data size and screen resolution when interactive frame rates are desired. Progressive rendering is a paradigm of displaying an approximative visual outcome early on, which iteratively and incrementally gets improved until convergence to the final result is reached. This approach maintains the interactivity of the system and allows the user to make decisions immediately, i.e., much earlier than the end of the computation process. We propose a method for progressive rendering of continuous scatterplots based on a Fourier representation. By iteratively advancing from low to high frequencies and inverting the spectrum representation after each iteration, a series of scatterplots converging to the final result is generated and rendered. We demonstrate that this convergence is monotonic and that the proposed approach is more efficient than state-of-the-art methods, i.e., we can faster produce high-quality approximations. We propose to embed this idea into a hybrid approach which allows balancing the trade-off between quality of the image appearing first and its computation time. The proposed algorithms were implemented on the GPU.

## Keywords

Scatterplot, Fourier transform, progressive rendering, unstructured volumetric data

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

## 1. INTRODUCTION

Most volumetric scalar fields, no matter whether they have been measured or simulated, are assumed to be continuous or, at least, piecewise continuous. Examples include fields of temperature, density, pressure, salinity, etc. Seldom, these scalar fields can be described analytically. In most cases, data values are acquired by sampling the scalar field at discrete locations in space. In order to reproduce their continuous nature, an interpolation method is involved.

Statistical properties of scalar fields can be explored via histograms. Pair wise relations of two such properties are usually represented in form of scatterplots. Multiple scatterplots can be investigated in form of a scatterplot matrix. All these techniques are well known and have been used for decades by operating on the values at the discrete sample locations. However, such an important property as continuity is being lost at this step.

Recently, new methods appeared which address the problem above by creating continuous representations of attribute spaces. They propose continuous histograms [CBB06, SSD\*08], continuous parallel coordinates [HW09, HBW11] and continuous scatterplots [BW08, BW09]. The idea behind these approaches is to perform a mapping of parts of the volume rather than just discrete samples when projecting to the visual domain. The distribution of attributes in the volume parts is reconstructed by means of interpolation.

Obviously, much more computational efforts are required to produce a continuous scatterplot when compared to its discrete analog. Therefore, there is a clear need for efficient algorithms showing high performance when applied to large datasets and high-resolution outputs. One approach to design an interactive tool is to exploit the idea of progressive rendering. This idea is based on an iterative generation of visualizations with increasing quality which converge to the final result. The gain is to provide the user with preliminary but still informative outcomes at interactive rates. Based on the preliminary result, the user can decide whether it is worth to wait for the final result or whether the interactive query shall be modified.

The essential properties of progressive rendering algorithm which we target in our research can be summarized as follows:

1. The key features of the result shall appear first.
2. The change of intermediate results when stepping from iteration to iteration shall vanish. In other words, the new contributions to the final image decrease in each iteration.
3. The result obtained at the previous step shall be effectively used when performing the next iteration, i.e., computations already made shall not be repeated.

Our work was inspired by the continuous scatterplots technique proposed by Heinrich et al. [HBW11]. In this algorithm, the physical volumetric domain is represented as a collection of Gaussian kernels centered at data samples. When mapping to the attribute domain (i.e., the domain spanned by the two dimensions of the scatterplot), each kernel appears as an elliptical splat called a *footprint*. The

geometry of a footprint is entirely defined by the parameters of a locally linearized mapping, which allows for the pre-computation of footprints in form of splat textures. To handle large datasets, Heinrich et al. propose to split the data into chunks, produce an individual plot for each portion of the data, and then iteratively combine the plots over several frames using alpha blending. The result is a progressively updated image which is based on the amount of data already processed. So, the work by Heinrich et al. uses the general idea of a progressive rendering. However, using their approach, it is not easy to fulfill the first two requirements formulated above. The result may be highly affected by the way, how the data are split into chunks. Sophisticated methods may help to make a wise decision, but they may also break the interactivity of the application.

Our main contribution is a novel approach for progressive rendering of continuous scatterplots based on their frequency representation (*spectrum*). This kind of representation is obtained by applying a Discrete Fourier Transform (DFT) to the scatterplot's density distribution. The key idea is as follows: Small-sized footprints have bad (slowly decaying) spectra but can be efficiently plotted directly into the attribute space density plot (continuous scatterplot). Large-sized footprints slow down the computations when blended in the attribute space, but they have nice frequency representations. Hence, we propose to split the physical space kernels into two classes: Those having small footprints (few pixels support) and those whose footprints have good Fourier representations. The former are plotted directly while the latter are processed iteratively, progressing from the lowest most meaningful modes to the highest frequencies with vanishing contributions.

Besides that, we extend the existing method to unstructured volumetric data and explore the performance and error dependence on most key parameters. We implemented the method by Heinrich et al. [HBW11] and our own algorithm entirely on the GPU (using CUDA) for better evaluation of their applicability and efficiency.

## 2. RELATED WORK

The idea of progressive image generation has been known for decades [BFGS86]. Progressive methods find their application in many visualization and computer graphics problems, e.g., ray tracing [PS89], global illumination [FP04], or volume rendering [CBPS06]. One of the most prominent examples is surface renderings where progressive meshes [Hop96] can be used in case of irregular meshes and subdivision surfaces in case of regular patches or semi-regular meshes. Also, isosurface extraction from progressively refined tetrahedral or pyramidal meshes (e.g., [LPD\*04]) and isosurface smoothing



[PB00] are common examples in volume visualization.

Spectral analysis is used to improve sampling of the rendering integral in volume rendering [BMW\*06]. A wide range of progressive spectral methods [SDS96], especially based on the Fast Fourier Transform (FFT), are very common in image processing, e.g., for image registration [RC96] or for characterization of brain fibers' shape [PEPM12].

Scatterplots are a well-known tool for multidimensional data visualization and analysis [EDF08]. Recently, an idea of continuous scatterplots was developed in a series of works by Bachthaler, Heinrich, and Weiskopf. Initially applied for mapping tetrahedra by using a linear interpolation of attributes within them [BW08], the approach was generalized to regular rectangular grids with an arbitrary interpolation method [BW09], where recursive subdivision of cells was exploited. Later, isotropic density functions were used to decompose the whole volume into a system of overlapping spheres [HBW11]. Splatting is performed for the generation of progressively sampled intermediate images which are then combined to produce the final continuous scatterplot. The progressive rendering component of the algorithm is based on partitioning the volumetric data into small chunks which are fast to operate. Intermediate images are produced for down-sampled or even freely re-sampled data. A generalization to continuous representations of projected spaces was recently proposed in [MFL13].

The idea of image space footprint computation of volumes has been developed by Westover [Wes90]. Feng et al. [FKLT10] use Gaussian footprints to visualize data samples uncertainty. Lehmann and Theisel [LT10] developed a method to find and highlight discontinuities in continuous scatterplots.

### 3. BACKGROUND

Before we describe our approach in detail, we would like to provide the respective background. For the construction of the continuous projections, we look into the settings of the involved spaces (a volumetric physical domain and a 2D attribute domain which is visualized). We discuss some technical aspects of the method by Heinrich et al. [HBW11] and, in particular, provide useful generalization by allowing the lengths of spatial kernels to vary from sample to sample.

#### 3.1 Basic Terms and Notations

Let  $\tau(\mathbf{x})$  be a multivariate multidimensional function with  $m$  variables defined over  $n$  dimensions, i.e.,  $\tau: \mathbb{R}^n \mapsto \mathbb{R}^m$ . It is sampled at position  $\{\mathbf{x}_i\} \subset X \subset \mathbb{R}^n$  mapping them to the set

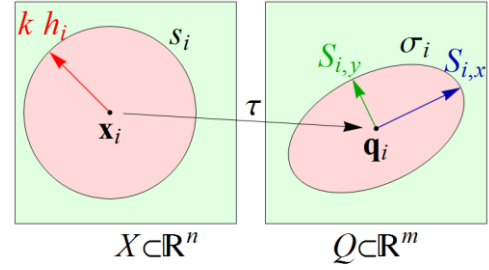


Figure 2: Physical volume  $X$  is mapped by  $\tau$  to the attribute domain  $Q$ . A spatial density  $s_i$  is defined in a spherical neighborhood of sample  $\mathbf{x}_i$ , which has an elliptical footprint in  $Q$ . Scalar function  $\sigma_i$  denotes density on the footprint.

$\{\mathbf{q}_i\} \subset Q \subset \mathbb{R}^m$ ,  $i=1, \dots, N$ . Figure 2 illustrates our notations. There exists a scalar non-negative function  $s_{\text{overall}}(\mathbf{x})$  defined on  $X$  which is called the *spatial density function*. It represents the importance of data in the volume and is usually set to be constant. For later consideration it is useful to approximate the density function by a weighted sum

$$s_{\text{overall}}(\mathbf{x}) = \sum_{i=1}^N w_i s_i(\mathbf{x}), \quad (1)$$

where every individual kernel  $s_i(\mathbf{x})$  is obtained from a compactly supported shape function  $s(\mathbf{x})$  by

$$s_i(\mathbf{x}) = s\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_i}\right),$$

where  $h_i$  defines a scaling radius of  $s_i(\mathbf{x})$ . In our work we focus on isotropic kernels such that  $s(\mathbf{x}) = s^r(\|\mathbf{x}\|)$ . Our goal is to find proper weights  $w_i$  to achieve  $s_{\text{overall}}(\mathbf{x}) \approx \text{const}$ . Since this condition simply means that all parts of domain  $X$  are equally important, the exact value of the constant does not matter. Then, if one integrates the relation (1) over the volume, one deduces

$$C_1 \cdot \text{Volume}(X) = \sum_{i=1}^N w_i \int_{\mathbb{R}^n} s_i(\mathbf{x}) d\mathbf{x} = C_2 \sum_{i=1}^N w_i h_i^n.$$

with constants  $C_1$  and  $C_2$ . This means that  $w_i h_i^n$  stand for volume fractions associated with samples. Therefore, one takes  $w_i = 1$  for all  $i$  to get  $s_{\text{overall}}(\mathbf{x}) \approx \text{const}$ .

In the following, we study the case  $n=3$  and  $m=2$  as the most practically important one, since it is common practice to investigate pairs of attributes defined over a volume by means of scatterplots.

Let  $X_i$  be the support of the function  $s_i(\mathbf{x})$ . The construction of a continuous scatterplot is based on additive mapping of all  $X_i$  weighted by  $w_i s_i(\mathbf{x})$  by means of  $\tau(\mathbf{x})$  to the attribute domain  $Q$ . Recall that discrete scatterplots map the data sample by sample resulting in the set  $\{\mathbf{q}_i\}$ . A continuous scatterplot is composed by a collection of images of  $X_i$ . It is shown in [HBW11] that spherical kernels  $s_i(\mathbf{x})$  result in elliptical footprints. Each footprint is equipped with a density distribution  $\sigma_i(\mathbf{q})$ , which represents a counterpart of the spatial density  $s_i(\mathbf{x})$ , see Figure 2. Hence, a continuous scatterplot  $\sigma_{\text{overall}}(\mathbf{q})$  is the direct sum of densities  $\sigma_i(\mathbf{q})$ .

The key relation between the two density functions is given by

$$\int_{X_0} s_{\text{overall}}(\mathbf{x}) d^n \mathbf{x} = \int_{Q_0} \sigma_{\text{overall}}(\mathbf{q}) d^m \mathbf{q}, \quad (2)$$

for any  $X_0 \subset X$  and  $Q_0 = \tau(X_0)$ . This condition uniquely defines  $\sigma_{\text{overall}}(\mathbf{q})$ . Equation (2) with  $X_0 \equiv X$  and  $Q_0 \equiv Q$  is called *total mass conservation*.

### 3.2 Direct Approach for Generation of Continuous Scatterplots

Following the method proposed in [HBW11], a spherical neighborhood of sample  $\mathbf{x}_i$  of radius  $kh_i$  is mapped as an elliptical splat centered at  $\mathbf{q}_i = (q_1(\mathbf{x}_i), q_2(\mathbf{x}_i))$  using a linear approximation of the mapping  $\tau$  in a neighborhood of sample  $\mathbf{x}_i$ . The density  $\sigma_i(\mathbf{q})$  is non-zero within this footprint. Besides its center, a footprint is uniquely defined by its extents  $S_{i,x}$  and  $S_{i,y}$  (semi-axes of the screen-space ellipse) and its rotation angle  $\theta_i$ . These parameters are defined as follows (cf. [Wes90, HBW11]):

$$S_{i,x} = kh_i \sqrt{\lambda_1}, \quad S_{i,y} = kh_i \sqrt{\lambda_2}, \quad (3)$$

$$\cos \theta_i = \frac{b}{\sqrt{b^2 + (\lambda_1 - a)^2}}, \quad (4)$$

where  $k$  is a global smoothing parameter discussed below,  $h_i$  is a local support size which may vary from sample to sample and reflect the samples' density in the neighborhood of the  $i$ -th instance,

$$\lambda_1 = \frac{a+c+e}{2}, \quad \lambda_2 = \frac{a+c-e}{2}, \quad e = \sqrt{(a-c)^2 + 4b^2}$$

$$a = \nabla q_1(\mathbf{x}_i) \cdot \nabla q_1(\mathbf{x}_i), \quad b = \nabla q_1(\mathbf{x}_i) \cdot \nabla q_2(\mathbf{x}_i),$$

$$c = \nabla q_2(\mathbf{x}_i) \cdot \nabla q_2(\mathbf{x}_i),$$

where  $\cdot$  stands for the scalar product. Note that the computation of  $\cos \theta_i$  can be numerically instable for small  $b$  when applying Equation (4) naively. If  $|b| < \epsilon$  for some threshold  $\epsilon > 0$ , the splat is not rotated, i.e.,  $\theta_i = 0$ . Thus, one can explicitly set  $\cos \theta_i = 1$ .

The user is allowed to change the value of parameter  $k$  which simultaneously scales the extents  $S_{i,x}$  and  $S_{i,y}$  of all footprints. Thus, the global smoothness of the continuous scatterplot can be controlled: For small values of  $k$  the result is almost a discrete plot, while large values of  $k$  make the resulting image blurred. This scaling is equivalent to the respective change of support sizes of spatial kernels  $s_i(\mathbf{x})$ . Here, their supports shrink to samples  $\mathbf{x}_i$  as  $k$  vanishes and unboundedly grow when  $k \rightarrow \infty$ . Note that for larger values of  $k$ , the number of locally overlapping spatial kernels increases.

All elliptical footprints are added as textures to the final image. In the following we refer to this procedure as a *direct (splatting) approach* in contrast to the progressive method proposed in this paper. After a normalization step which makes the range of the accumulated density  $\sigma_{\text{overall}}(\mathbf{q})$  equal to  $[0,1]$ , a transfer function can be applied. The main bottleneck of the method is related to the rendering of large splats (relative to the screen resolution), since their contributions to many pixels have to be computed. This situation occurs in particular when producing high-resolution images, when zooming into a smaller region of the plot, or when choosing high values of the global smoothing parameter  $k$ .

### 3.3 Progressive Approach

To overcome the issue of insufficient efficiency of the method for interactive visual analyses, a progressive splatting approach was proposed in [HBW11]. When splitting the input data into small portions and operating on them separately, the number of operations per chunk is reduced. It makes it possible to generate images for each chunk faster.

Gradual accumulation of the generated images results in a progressively changing continuous scatterplot. However, every intermediate result intrinsically depends on which part of the data is already processed and which not. In other words, it depends on the order in which the chunks are accumulated. To illustrate this effect we generated a "Tornado" dataset, courtesy of [CM93], sampled at  $40^3$  regularly distributed nodes. Velocity magnitude and the  $z$ -component of the velocity field are taken

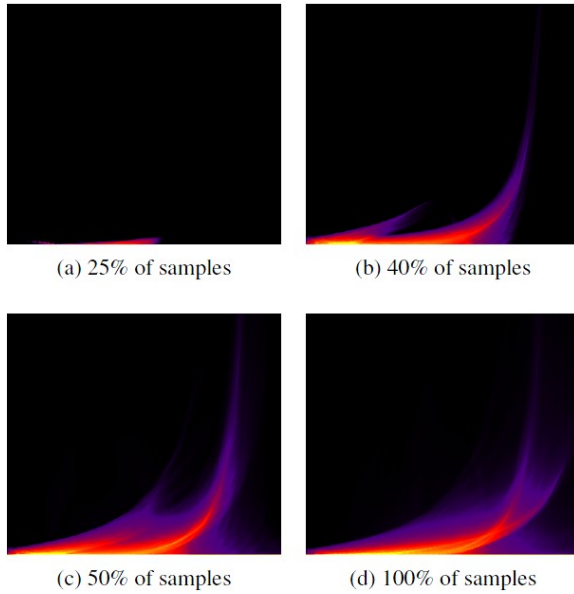


Figure 3: Dependence of intermediate images in progressive splatting on the fraction of processed data ("Tornado" dataset with  $40^3$  gridded nodes).

as dimensions of the scatterplot. Intermediate results after operating 25%, 40% and 50% of samples as well as the final result are shown in Figure 3. All four images are qualitatively very different, so that there is no monotonic behavior of the results along iterations. Hence, this unpredictability of the intermediate results will make the user wait until all computations are done, which destroys the essence of progressive rendering (see Requirements 1 and 2 formulated in the Introduction). Of course, the intermediate images depend on how the chunks of data have been generated and a different choice may have produced qualitatively better approximations early on, but the issue of not knowing whether the intermediate results reflect the final result well is apparent.

We also note that the normalization of the density, which is necessary before a transfer function can be applied, is very sensitive to the skipping of data portions. In fact, skipping a few samples contributing to the pixel with the highest density obviously changes the normalization factor, which affects the appearance of the whole picture, although the transfer function remains the same.

The situation becomes even worse when dealing with unstructured spatial datasets, which are discussed in the next section. The reason is that there is no standard way of ordering data in memory and splitting data into chunks can be absolutely arbitrary. Note that even if the data are regularly sampled, after a zooming operation, the data to be displayed generally do not belong to any spatially regular structure. These considerations motivated us to

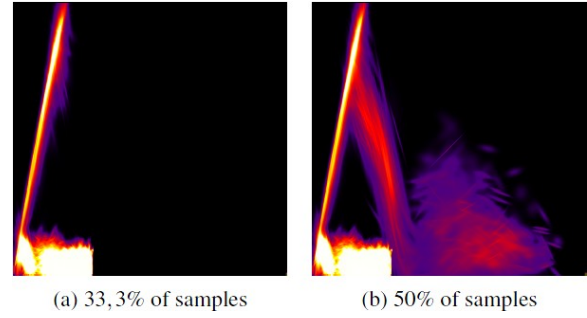


Figure 4: Dependence of intermediate images in progressive splatting on the fraction of processed data ("two White Dwarfs" dataset with 24k unstructured samples).

design an approach, where all samples are handled uniformly and simultaneously and where no particular spatial arrangement of the samples is assumed.

Our solution to the issue above relies on the spectral representation of continuous scatterplots. Instead of splitting the data, the progressive part of our approach is based on a consecutive computation of the Fourier frequencies advancing from the lowest most contributing modes to the highest less important ones. In the following section we show that large splats have a nice (rapidly decaying) representation in the frequency domain that makes it possible to significantly speed up the construction of continuous scatterplots.

We note that spatially unorganized data can be stored in a file in arbitrary order. Thus, when applying the progressive splatting idea from [HBW11], the distribution of samples among chunks of data is very accidental. This fact leads to the effect shown in Figure 4. Shown is a dataset representing an astrophysical simulation of a binary system consisting of two White Dwarfs. The dimensions of scatterplot are the internal energy values and temperature at samples' positions. Images for 33.3% and 50% of the samples provide a poor approximation to the final result shown in Figure 5 (e).

### 3.4 Spatially Unstructured Data

We generalize the approach of Heinrich et al. [HBW11] by allowing the local scales to vary from sample to sample. It makes it possible to apply the direct approach to irregularly sampled data. In order to define an individual scale  $h_i$ , we estimate the local density of samples: In the regions, where samples are dense,  $h_i$  is small, while sparsely distributed  $\mathbf{x}_i$  yield large  $h_i$ . Only the relative variation of scales is important, since the absolute magnitude can be changed by adapting parameter  $k$ .

For regular data, the samples' density is constant everywhere, hence, all  $h_i$  are equal.

The ability to handle irregular data is very valuable for practical applications. For instance, various scientific measurement techniques and particle-based numerical methods may produce large multivariate unstructured spatial datasets which are to be visualized and analyzed. Of course, it is always possible to re-sample unstructured data regularly, but this comes at the cost of introducing interpolation errors and losing spatial adaptivity which may affect the result. Therefore, it is advantageous to operate on the original data.

#### 4. SPECTRAL REPRESENTATION OF CONTINUOUS SCATTERPLOTS

For our purposes, it is important that density  $\sigma_i(\mathbf{q})$  in the footprint domain can be computed from some template kernel function  $\sigma(\mathbf{q})$  via an affine transformation composed by scaling, rotation, and translation, i.e.,

$$\sigma_i(\mathbf{q}) = \alpha_i \sigma(L_i(\mathbf{q} - \mathbf{q}_i)),$$

$$L_i = \begin{pmatrix} S_{i,x}^{-1} & 0 \\ 0 & S_{i,y}^{-1} \end{pmatrix} \begin{pmatrix} \cos \theta_i & \sin \theta_i \\ -\sin \theta_i & \cos \theta_i \end{pmatrix}.$$

In other words, to compute  $\sigma_i(\mathbf{q})$  at some location, one has to perform translation by  $\mathbf{q}_i$ , rotation by angle  $\theta_i$ , scaling by  $(S_{i,x}, S_{i,y})$ , and then evaluate the shape-function  $\sigma$  at the obtained location. This is just an analytical expression of the splatting, where a pre-computed texture containing values of  $\sigma(\mathbf{q})$  is placed at the right position in the frame with right scaling and right orientation resulting in the footprint  $\sigma_i(\mathbf{q})$ . The coefficient  $\alpha_i = w_i h_i^3 \cdot S_{i,x}^{-1} \cdot S_{i,y}^{-1}$  serves to conserve the mass associated with the  $i$ -th sample:

$$\int_Q \sigma_i(\mathbf{q}) d^2 \mathbf{q} = \alpha_i \int_Q \sigma(L_i(\mathbf{q} - \mathbf{q}_i)) d^2 \mathbf{q}$$

$$= \frac{\alpha_i}{|L_i|} \int_Q \sigma(\mathbf{p}) d^2 \mathbf{p} = w_i h_i^3 \int_X s(\mathbf{y}) d^3 \mathbf{y} = w_i \int_X s_i(\mathbf{x}) d^3 \mathbf{x}.$$

Here we assumed that the  $L_i$ -norms of template functions  $s$  and  $\sigma$  are equal and used  $|L_i| = S_{i,x}^{-1} \cdot S_{i,y}^{-1}$ . Summation over  $i$  provides the total conservation of mass (2).

The final density distribution is given as a composition of individual contributions of all samples, i.e.,

$$I(q_1, q_2) = \sum_{i=1}^N \sigma_i(q_1, q_2).$$

Here and further we use  $I \equiv \sigma_{\text{overall}}$  for short. Now we examine the density function in the frequency domain. By the linearity property, the Fourier transform of  $I(q_1, q_2)$  reads

$$\mathcal{F}[I](u, v) = \sum_{i=1}^N \mathcal{F}[\sigma_i](u, v).$$

It is known that scaling, rotation, and translation operators become scaling, rotation and modulation after the Fourier transform, correspondingly. In particular,  $\mathcal{F}[f(A\mathbf{x})](\mathbf{q}) = |A^{-1}| \cdot \mathcal{F}[f(\mathbf{x})](A^{-1}\mathbf{q})$ . Thus,

$$\mathcal{F}[\sigma_i](u, v) = e^{i(uq_1(\mathbf{x}_i) + vq_2(\mathbf{x}_i))} \alpha_i |L_i^{-T}| \cdot \mathcal{F}[\sigma](L_i^{-T}(u, v)),$$

where  $-T$  denotes inversion and transposition, with

$$L_i^{-T} = \begin{pmatrix} S_{i,x} & 0 \\ 0 & S_{i,y} \end{pmatrix} \begin{pmatrix} \cos \theta_i & \sin \theta_i \\ -\sin \theta_i & \cos \theta_i \end{pmatrix}.$$

Finally,

$$\mathcal{F}[I](u, v) = \sum_{i=1}^N w_i h_i^3 e^{i(uq_1(\mathbf{x}_i) + vq_2(\mathbf{x}_i))} \mathcal{F}[\sigma](L_i^{-T}(u, v)). \quad (5)$$

When the spectral representation  $\mathcal{F}[I]$  is computed, one can apply the inverse Fourier transform to obtain the continuous scatterplot. Practically, we use the discrete Fourier transform rather than the Fourier transform itself. This approximation introduces errors which vanish if the resolution of textures (number of frequencies taken into account) grows. We also need to comment on the periodicity of the discrete Fourier transform: Footprints located close to the boundary of the computed region may have their parts appearing at the opposite part of the boundary instead of being cut. To overcome this effect, the region should be extended to reserve additional space for such footprints.

#### 5. PROGRESSIVE ALGORITHM

Our complete computation of spectral representation  $\mathcal{F}[I]$  is much slower than the standard splatting-based technique for continuous scatterplot generation. Obviously, every footprint has usually non-zero contributions to all frequencies  $(u, v)$ , whereas the splat may have very small size. Thus, if the resolution of the screen is  $R_u \times R_v$ , complexity of  $\mathcal{F}[I]$  computation is  $N \times R_u \times R_v$ . Recall that this representation has to be transformed by the inverse FFT and not the full resolution can be shown, since some border region of the texture is reserved to eliminate the periodicity effect of DFT. However, there is a strong side of  $\mathcal{F}[I]$ .

It is well-known that the decay of Fourier coefficients depends on the smoothness of a function:

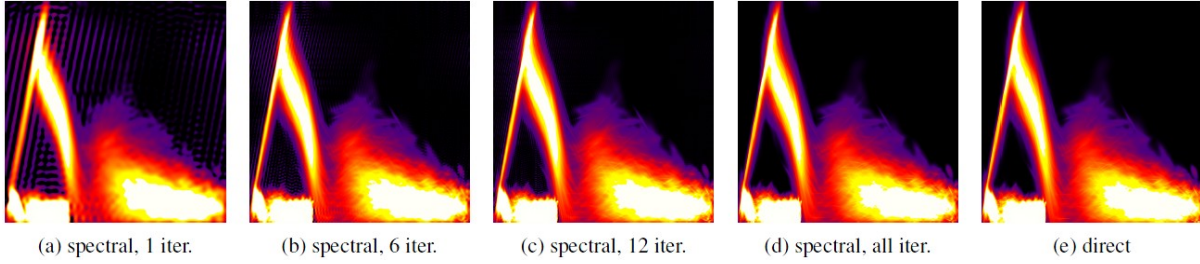


Figure 5: Continuous scatterplots of merging White Dwarfs dataset. When all footprints are progressively computed using spectral method, small splats cause noise, which is visible at first iterations and disappears later. After all iterations are completed, the result is identical to one obtained by the direct approach, i.e., when all elliptical footprints are blended as textures.

The smoother the kernel  $\sigma(\mathbf{q})$ , the faster the decay of  $|\mathcal{F}[\sigma](u,v)|$ . In fact, for many footprints, high frequencies do not significantly affect the result and therefore may be neglected. Based on this observation, we propose the following progressive algorithm for continuous scatterplot construction: Evaluate  $\mathcal{F}[I](u,v)$  for some low frequencies, perform the inverse FFT and render the result as the first approximation to the final picture. In subsequent iterations higher frequencies are added to the Fourier representation of the scatterplot and the rendered picture is updated after performing the inverse FFT. Since high frequencies of  $\mathcal{F}[I]$  computed at later iterations are composed by vanishing contributions from individual splats, later iterations have generally less impact on the final result for smooth kernels. Thus, Requirements 1-3 in the Introduction are fulfilled. The number of frequencies computed in each iteration depends on the hardware performance and the data size. It can be fixed or vary from one step to another.

## 6. HYBRID ALGORITHM

Large footprints have fast decaying Fourier spectra and therefore can be very efficiently and with high precision represented by a few lowest frequencies. Small footprints can be fast rendered directly to the scatterplot. Here "small" means that at least one of the extents  $S_{i,x}$  and  $S_{i,y}$  is less than a prescribed threshold. To profit from the best of both worlds, we divide all footprints into two groups according to their size and operate accordingly. The grouping is based only on values  $S_{i,x}$  and  $S_{i,y}$ , i.e., no additional computations are needed. Moreover, the critical size of a splat below which it is labeled as small, depends only on the screen resolution. Since parameter  $k$  scales all splats, its value affects the grouping.

The idea of the hybrid approach is illustrated in Figure 1. We used the "Bucky Ball" dataset (courtesy AVS, USA) with  $32^3$  gridded samples. Small and large splats rendered by means of the direct method

and their spectral representation, respectively, see Figure 1 (a) and (b). Both plots are combined to produce the final result (c) which is very close to the scatterplot computed by direct splatting (d).

## 7. RESULTS

All numerical tests were performed on a PC with graphic card NVIDIA GTX 680 and implemented in CUDA. We used the same color scheme to render all continuous representations of projections, see Figure 1 (e). The texture size used for computation is  $1024^2$  pixels. We reserved 10% for the border, such that the size of the rendered texture is  $921^2$  pixels. Computation of the whole spectral representation requires  $1024^2 / 4096 = 256$  iterations, where parameter 4096 is chosen to achieve a high occupancy of the GPU. If other is not specified,  $s_{\text{overall}}(\mathbf{x}) = \text{const}$  is used. Other information about parameters and datasets is shown in Table 1.

dataset	k	threshold	Total number of points	Number of large splats
Bucky Ball	2.0	$0.02 * R_u$	32768	13986
White Dwarfs	1.6	$0.008 * R_u$	24149	21676
Tornado	0.6	$0.01 * R_u$	262144	82856

Table 1: Parameters and thresholds for all experiments. Shown are values of the global smoothing parameter  $k$ , the threshold that defines which splats are labeled as small, the size of dataset, and the number of splats computed by the spectral method for the given threshold.

First, we demonstrate the convergence of the pure spectral approach. The dataset represents an astrophysical simulation of two merging White



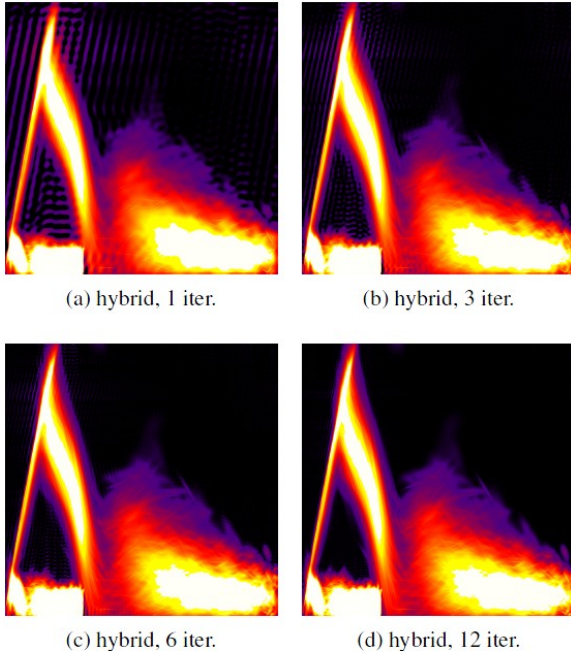


Figure 6: Hybrid approach applied to the merging White Dwarfs dataset.

Dwarfs. The simulation was executed by means of the Smoothed Particle Hydrodynamics method [Luc77, GM77]. The data includes unstructured nodes' positions  $\mathbf{x}_i$ , lengths  $h_i$ , several scalar attributes and their gradients. Dimensions of the scatterplot are internal energy and temperature fields in horizontal and vertical directions, correspondingly. Results for  $s_{\text{overall}}(\mathbf{x})$  to be equal to the density field are shown in Figure 5. Small splats have poor spectral representation and, thus, serve as a source of noise. This noise is eliminated during progressive computation of Fourier magnitudes. After all iterations are completed, the result is identical to one obtained by the direct method. When the hybrid approach is applied to the same dataset, the initial level of noise is much lower, see Figure 6.

Next, we used the "Tornado" dataset, courtesy of [CM93]. It is given as an analytical function describing a velocity profile in a volume. We sampled the velocity field at  $64^3$  random uniformly distributed locations. Results obtained by the hybrid approach are presented in Figure 7. Velocity components in  $x$  and  $y$  directions serve as the two axes of the scatterplots. We intentionally chose a relatively small  $k$  to demonstrate the effect of detailization during progressive rendering. Small global smoothness makes the sizes of footprints small, thus, even some individual splats remain distinguishable in the continuous scatterplot. The first iteration of the hybrid approach results in a slightly smeared image with more and more details appearing at later steps.

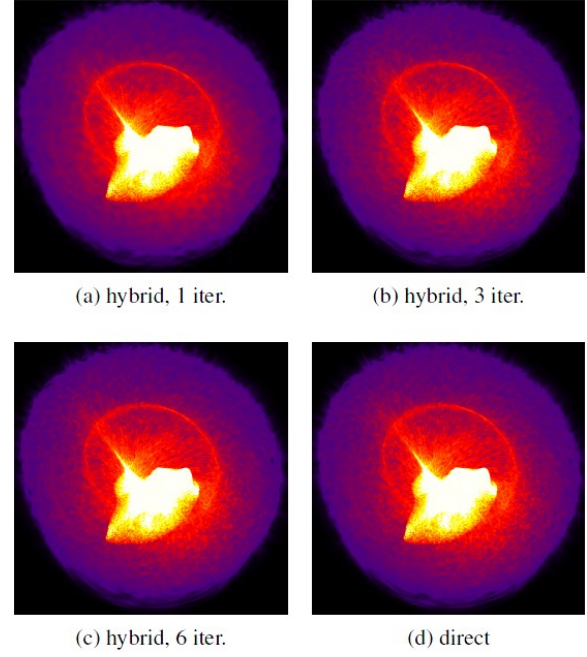


Figure 7: Hybrid approach applied to the "Tornado" dataset.

Computational times are listed in Table 2. Both the spectral and the hybrid approaches deliver preliminary results much earlier than the direct splatting method. Though, the first iteration of the hybrid method takes a longer time than one iteration of the spectral method, its later iterations are faster and the quality of the result is higher. The accompanying video shows a comparison of the quality of the results obtained with spectral and hybrid approach. Note that the first iterations of the video are played in slow-motion to have enough time to observe the intermediate images.

To measure the error of the proposed methods at  $j$ -th iteration, we computed the root-mean-square error

$$L_j = \left( \frac{1}{R_u \cdot R_v} \sum_{p_x=1}^{R_u} \sum_{p_y=1}^{R_v} (\bar{\sigma}(p_x, p_y) - \bar{\sigma}_{\text{direct}}(p_x, p_y))^2 \right)^{1/2},$$

where  $p_x$  and  $p_y$  stands for pixel indices,  $\bar{\sigma}_{\text{direct}}$  is the density computed by the direct approach,  $\bar{\sigma}$  can be computed either by the spectral or by the hybrid method, and the bars denote that the densities are normalized (individually). Results are shown in Figure 8. It is evident that the error is gradually improved along the progressive computations. The error of the hybrid method after first iteration is significantly less than the analogous error of the spectral one. Moreover, only a few first iterations of the hybrid method suffice to closely approach a stable state.

The hybrid approach has better error behavior though the first step takes more computation time.

dataset	direct	Spectral, per iter.	Hybrid, 1st iter.	Hybrid, from 2nd iter.
Bucky Ball	3942 ms	119 ms	435 ms	48 ms
White Dwarfs	895 ms	95 ms	269 ms	76 ms
Tornado	1254 ms	591 ms	282 ms	203 ms

Table 2: Computation times for construction of continuous scatterplots using direct approach, pure spectral representation and the hybrid algorithm (all methods are implemented on GPU).

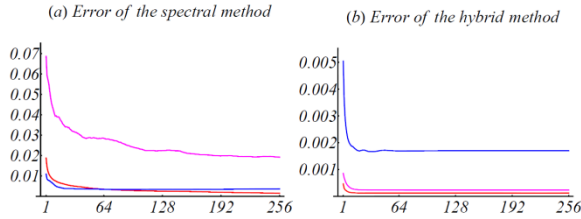


Figure 8: Error behavior along the iterative construction of continuous scatterplots. The plots present the root-mean-square errors of the spectral algorithm and the hybrid method. Results for the "Bucky Ball" (red), two White Dwarfs (blue) and the "Tornado" (magenta) datasets are shown. The error of the hybrid approach applied to the "Bucky Ball" dataset (red line in (b)) is multiplied by a factor of  $10^3$  to be visible.

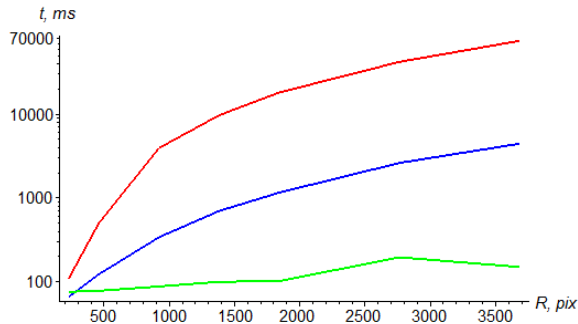


Figure 9: Plots of calculation times of appearance of the first meaningful image against output resolution (pixels) computed for "Bucky Ball" dataset. Results for the direct splatting (red), the hybrid method (blue) and the spectral approach (green) are shown.

As follows from the data in Table 2, almost 7 iterations of the spectral method can be finished before the first iteration of the hybrid method is completed. However, the initial error of the hybrid approach is lower than the error of the spectral algorithm after 7 iterations and decreases faster in the consequent steps. This holds for all datasets. By the time, when the first result of the hybrid approach is delivered only 11% ("Bucky Ball"), 30% ("two White Dwarfs") and 22% ("Tornado") of samples can

be processed by the direct method, which is insufficient for performing a reliable analysis.

The benefit of our approach becomes extremely significant when producing high-resolution outputs. Figure 9 demonstrates the dependence of calculation times for the first meaningful picture on the output resolution. In particular, the speed-up of 500 times is achieved for  $R_u = 3686$  pixels. Note that the values of time (vertical axis) are scaled logarithmically. Non-monotonic behavior of the green line is related to higher efficiency of FFT when  $R$  is a power of 2.

## 8. CONCLUSION

We have seen that when producing high-resolution continuous scatterplots, when dealing with highly adaptive sampling in physical space, when zooming into a region of interest on a scatterplot, or when increasing the global smoothness of the result, sizes of some footprints become large up to the range of the screen size. In such situation, a direct accumulation of the splats in the final image gets extremely slow and significantly affects the overall performance, which hinders interactivity. Our method allows overcoming this drawback by the effective use of the spectral representation of large footprints.

The proposed progressive rendering approach fulfills all three requirements formulated in the Introduction section. First, small splats are directly aggregated into the density texture at the very first step of the algorithm. These splats usually have highest density values and thus affect the overall picture most, since they mostly determine the range of the transfer function being applied. Second, due to decay of the Fourier coefficients of a smooth function, each next iteration in the Fourier domain will have less impact than the previous step. Thus, changes in the displayed picture lessen over the iteration steps and stabilize to the final state quite rapidly, which was shown in a number of tests. Finally, only missing modes of spectral representation are added in each subsequent step, so that no re-computation of any part of earlier obtained results is needed.

We have been able to significantly speed up the appearance of the first result of the hybrid method when compared to the full-resolution direct splatting approach. In our tests, the speed-up varied from 3.3 times to two orders of magnitude depending on the resolution of the output. This allows for computation times suitable for interactive analysis, e.g., in scatterplot matrices.

## 9. ACKNOWLEDGMENTS

The authors wish to thank Marius Dan and Stephan Rosswog for sharing datasets. This work was supported in part by a DFG grant LI 1530/6-2.

## 10. REFERENCES

- [BFGS86] Bergman L., Fuchs H., Grant E., Spach S.: Image rendering by adaptive refinement. *SIGGRAPH Comput. Graph.* 20, 4 (1986), 29–37.
- [BMW\*06] Bergner S., Möller T., Weiskopf D., Muraki D.: A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 1353–1360.
- [BW08] Bachthaler S., Weiskopf D.: Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization)* 14, 6 (2008), 1428–1435.
- [BW09] Bachthaler S., Weiskopf D.: Efficient and adaptive rendering of 2-d continuous scatterplots. *Comput. Graph. Forum (Proc. Eurovis 09)* 28, 3 (2009), 743 – 750.
- [CBB06] Carr H., Duffy B., Denby B.: On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 1259–1266.
- [CBPS06] Callahan S. P., Bavoil L., Pascucci V., Silva C. T.: Progressive volume rendering of large unstructured grids. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 1307–1314.
- [CM93] Crawfis R., Max N.: Texture splats for 3D vector and scalar field visualization. In *Proceedings Visualization '93* (1993), IEEE CS Press, 261–266.
- [EDF08] Elmqvist N., Dragicevic P., Fekete J.-D.: Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), 1141–1148.
- [FKLT10] Feng D., Kwock L., Lee Y., Taylor R.: Matching visual saliency to confidence in plots of uncertain data. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 980–989.
- [FP04] Farrugia J.-P., Péroche B.: A progressive rendering algorithm using an adaptive perceptually based image metric. In *Eurographics conference proceedings* (2004).
- [GM77] Gingold R. A., Monaghan J. J.: Smoothed particle hydrodynamics — theory and application to non-spherical stars. *Mon. Not. Roy. Astron. Soc.* 181 (1977), 375–389.
- [HBW11] Heinrich J., Bachthaler S., Weiskopf D.: Progressive splatting of continuous scatterplots and parallel coordinates. *Comput. Graph. Forum* 30, 3 (2011), 653–662.
- [Hop96] Hoppe H.: Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), SIGGRAPH '96, ACM, 99–108.
- [HW09] Heinrich J., Weiskopf D.: Continuous parallel co-ordinates. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1531–1538.
- [LPD\*04] Linsen L., Pascucci V., Duchaineau M. A., Hamann B., Joy K.: Wavelet-based multiresolution with  $n$ th-root-of-2 subdivision. *Journal on Computing special edition* (2004).
- [LT10] Lehmann D. J., Theisel H.: Discontinuities in continuous scatter plots. *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), 1291–1300.
- [Luc77] Lucy L. B.: A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal* 82 (1977), 1013–1024.
- [MFL13] Molchanov V., Fofonov A., Linsen L.: Continuous Representation of Projected Attribute Spaces of Multifields over Any Spatial Sampling. *Comput. Graph. Forum (Proc. Eurovis 13)* (2013), to appear.
- [PB00] Pascucci V., Bajaj C. L.: Time critical isosurface refinement and smoothing. In *Proceedings of the 2000 IEEE symposium on Volume visualization* (2000), VVS '00, ACM, 33–42.
- [PEPM12] Poco J., Eler D. M., Paulovich F. V., Minghim R.: Employing 2d projections for fast visual exploration of large fiber tracking data. *Comp. Graph. Forum* 31, (2012), 1075–1084.
- [PS89] Painter J., Sloan K.: Antialiased ray tracing by adaptive progressive refinement. *SIGGRAPH Comput. Graph.* 23, 3 (1989), 281–288.
- [RC96] Reddy B. S., Chatterji B. N.: An FFT-based technique for translation, rotation, and scale-invariant image registration. *Trans. Img. Proc.* 5, 8 (1996), 1266–1271.
- [SDS96] Stollnitz E. J., Deroose T. D., Salesin D. H.: *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
- [SSD\*08] Scheidegger C., Schreiner J., Duffy B., Carr H., Silva C.: Revisiting histograms and isosurface statistics. *Visualization and Computer Graphics, IEEE Transactions on* 14, 6 (2008), 1659 – 1666.
- [Wes90] Westover L.: Footprint evaluation for volume rendering. In *Computer Graphics* (1990), 367–376.



# Novel Patterns and Methods for Zooming Camera Calibration

Andrea Pennisi, Domenico Bloisi, Claudio Gaz, Luca Iocchi, Daniele Nardi

Department of Computer, Control, and Management Engineering

Sapienza University of Rome

via Ariosto 25

00185, Rome, Italy

{pennisi,bloisi,gaz,iocchi,nardi}@dis.uniroma1.it

## ABSTRACT

Camera calibration is a necessary step in order to develop applications that need to establish a relationship between image pixels and real world points. The goal of camera calibration is to estimate the extrinsic and intrinsic camera parameters. Usually, for non-zooming cameras, the calibration is carried out by using a grid pattern of known dimensions (e.g., a chessboard). However, for cameras with zoom functions, the use of a grid pattern only is not sufficient, because the calibration has to be effective at multiple zoom levels and some features (e.g., corners) could not be detectable. In this paper, a calibration method based on two novel calibration patterns, specifically designed for zooming cameras, is presented. The first pattern, called in-lab pattern, is designed for intrinsic parameter recovery, while the second one, called on-field pattern, is conceived for extrinsic parameter estimation. As an application example, on-line virtual advertising in sport events, where the objective is to insert virtual advertising images into live or pre-recorded television shows, is considered. A quantitative experimental evaluation shows an increase of performance with respect to the use of standard calibration routines considering both re-projection accuracy and calibration time.

## Keywords

Virtual advertisement, zooming camera calibration, augmented reality.

## 1 INTRODUCTION

Having an accurate calibration is crucial in many computer vision applications, ranging from automatic video surveillance to augmented reality. In order to calibrate a non-zooming camera, a grid pattern with known dimensions can be used. Usually a chessboard with black and white squares is chosen, allowing to obtain a good correspondence between image and world points [Har04a].

However, when the camera has the capability of zooming, the use of a chessboard only is not sufficient. Indeed, since the calibration has to be carried out at different zoom levels, the accuracy of the correspondence between scene and image points can decrease, especially at high zoom levels.

In particular, when shooting sport events a lot of changes in the zoom levels are required to capture objects that can be at different distances from the

camera. Using standard multi-level calibration techniques (e.g., [AIA06a, May97a, Stu97a]) can introduce re-projecting errors. Indeed, since the calibration is repeated at different zoom levels, each calibration is influenced by the error introduced at the previous zoom level, thus inevitably increasing the final re-projecting error.

In this paper a calibration method based on two novel calibration patterns, specifically designed for variable-zoom cameras, is presented. The two patterns allow to calculate a set of calibration parameters that do not depend on the camera zoom level. The first pattern is designed for computing an accurate calculation of the camera intrinsic parameters even at high zoom levels, while the second one is conceived for reducing the time needed to calculate the extrinsic parameters of the camera and for making easier and faster the on-field calibration process.

The method has been applied to a real system for adding virtual advertisement in live sport events that is currently used by Duel TV S.p.A.<sup>1</sup>, a company that provides real time virtual advertisement services for sport events. In order to quantitatively evaluate the performance of our calibration approach, it is compared with

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

<sup>1</sup> [www.dueltv.com](http://www.dueltv.com)

the previous one used by the company, taking into account both the precision in placing the virtual billboards and the calibration time.

The remainder of the paper is organized as follows. After the analysis of related work in Section 2, Section 3 describes the two novel patterns and the calibration routines to estimate the camera parameters. A real-world application scenario is described in Section 4, while Section 5 shows experimental results. Conclusions are drawn in Section 6.

## 2 RELATED WORK

Several solutions have been proposed in literature to find methods that aim at achieving an accurate camera calibration (see [Sal02a] for a detailed survey). The use of zooming cameras adds a further level of complexity, since a high accuracy may be maintained over different levels of zoom. A set of calibration methods for zooming cameras are discussed in the following.

Al-Ajlouni and Fraser in [AlA06a] propose a zoom-dependent calibration process, whereby the traditional image coordinate correction model for camera interior orientation and lens distortion is expressed as a function of the nominal zoom focal length written into the EXIF header of the image file. This removes the requirement of using fixed zoom/focus settings for the images forming the photogrammetric network. However, the authors assume that the computation of the empirically determined Z-D calibration parameters is unlikely to be required too frequently, that is a too strong assumption for real-world settings (e.g., sport events).

Maybank and Faugeras in [May97a] estimated the intrinsic parameters of the camera considering that some constraints based on the epipolar geometry of two views are related to the rigidity of camera movements. To compute the parameters of the epipolar equation, they select some points in the current scene and track them in the image while moving the camera with a random motion. The main drawback of this method is the loss of accuracy generated by the manual selection of the points.

Strum in [Stu97a] proposes to set unknown parameters in order to solve the problems in [May97a]. An interdependence model of intrinsic parameters is computed in a pre-calibration phase. In such a way, the calibration is reduced to the estimation of only one parameter from which all the others depend. However, it is not clear if this method is effective for imaging systems where the interdependence of parameters follows a more complicated model with respect to the one described by the author.

Hyunwoo and Ki Sang in [Kim00a] propose a method for simplifying the computation of the camera intrinsic

parameters. The authors are able to overcome the degenerate configurations and to get a closed form solution. They introduce also a non-linear algorithm that adjusts both camera parameters and inter-image homography, thus a more accurate image registration becomes possible. However, even if such a method is useful for moderate camera rotations and zooming variations, there is an increase of failures for wider camera movements.

Oh *et al.* in [Oh09a] present a method to calibrate pan-tilt-zoom-focus cameras using both a pattern- and a rotation-based calibration approach. The method is composed by two separate procedures for calibrating zoom and focus respectively. The zoom calibration is based on the detection of a set of known patterns (chessboards) and it is carried out at different zoom levels. The focus calibration is obtained considering first the lowest focus value of the camera, and then, for the remaining focus values, by applying an automatic procedure, that takes into account the previous calculated zoom calibrations. This approach has good results for real cameras with translation offsets, but it generates incorrect results at high-zoom levels. Moreover, it uses 10 calibration patterns and the calibration must be repeated for all zoom levels.

Sinha *et al.* in [Sin06a] deal with the problem of estimating the parameters of the calibration model for active pan-tilt-zoom cameras. The camera intrinsic parameters are estimated over its full range of pan, tilt, and zoom by computing homographies between images acquired by a rotating and zooming camera. The calibration algorithm also computes accurate calibrated panoramas at multiple levels of detail. The main weakness of this method is the need of re-estimating the calibration parameters every time the camera moves.

Sarkis *et al.* in [Sarkis07a] propose a least-square approach to model the variation of internal parameters as a function of focus and zoom. This approach is able to increase the accuracy in modelling the intrinsic parameters of a zoom lens camera system and to decrease the pixel re-projection error. However, the method is sensitive only to large variations of focus and zoom values.

Finally, Agapito *et al.* in [Aga01a] use an homography to compute intrinsic parameters, assuming that the aspect-ratio and the principal point are fixed in time, while the focal length changes as the camera moves. They assume also that the principal point is the best of the aspect ratio and it can be considered as the image center. Given the principal point, it becomes simpler to auto-calibrate the camera. The main drawback of this method is the need to find the principal point, that is a non-trivial task.

All of the above discussed methods have limitations that make them not robust to zoom variations and/or not easy to set up. In the next sections we will describe

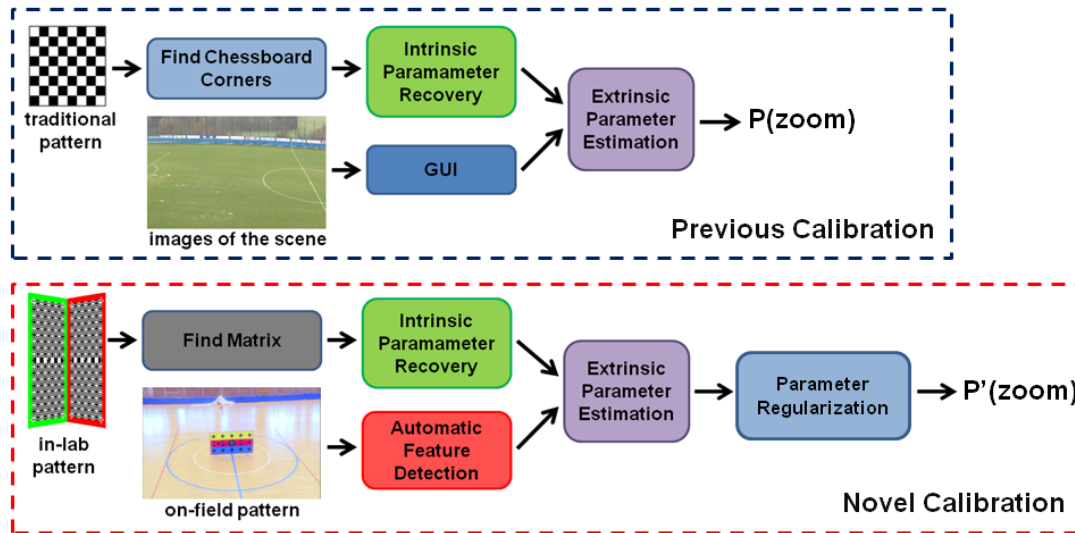


Figure 1: The proposed approach. With respect to common calibration methods (in this figure the one previously used by the company is reported), in the novel method two different calibration patterns are used for in-lab and on-field calibration, respectively.

our approach that aims at providing an easier calibration procedure that is robust to zoom variations.

### 3 OVERALL APPROACH

In order to achieve the two goals of camera calibration (namely, intrinsic parameter recovery and extrinsic parameter estimation) [Har04a], we propose the use of two novel calibration patterns. With respect to common calibration methods, the proposed patterns allow to obtain a more accurate intrinsic parameter calculation and an automatic extrinsic parameter estimation, with the possibility of regularizing camera parameters in function of the zoom levels (see Fig. 1).

The first pattern, called *in-lab pattern*, is used to recover the intrinsic camera parameters. The *Find Matrix* module is responsible for detecting the features in the pattern that are sent as input to the *Intrinsic Parameter Recovery* module. Such a module uses standard routines [Sir04a] to calculate the intrinsic parameters of the camera. A second pattern, called *on-field pattern*, allows to estimate the extrinsic parameters in a fully automatic way thanks to the *Automatic Feature Detection* module. The introduction of the on-field pattern replaces the graphical user interface (GUI) that was used in the previous calibration method to associate image points in the captured frame to known world coordinates (e.g., soccer field line). A novelty is also represented by the *Parameter Regularization* module, that aims at refining the calibration parameters at different zoom levels.

#### 3.1 In-lab Pattern

To achieve an accurate calibration even in presence of high zoom levels, we focussed our analysis on the calibration pattern. First, we analysed the results obtained

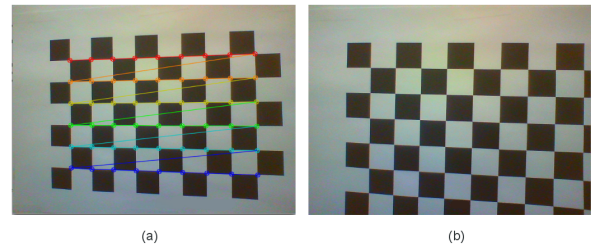


Figure 2: Traditional calibration using a chessboard pattern. a) The OpenCV routines can detect all the chessboard corners. b) The corners cannot be associated to the corresponding points on the chessboard if the pattern is not completely captured.

using a traditional chessboard (with black and white squares) at different zoom levels. To compute the results, we used the methods provided by the OpenCV library (version 2.4.4) [Ope13a].

We observed that if the chessboard is entirely captured, then the calibration accuracy is good (Fig. 2a). However, for high zoom levels, it can happen that not all the corners are captured and, as a consequence, it is impossible to carry out the calibration, because the found corners cannot be associated with the corresponding ones on the chessboard pattern (Fig. 2b). Therefore, a pattern allowing for a non-ambiguous association even if it is not completely captured should be used.

It is worth noting that, in order to calibrate a zooming camera, the focus parameter has to be maintained stable and, for this reason, increasing the zoom level can generate blurred images. Since the squares are effective features to be detected at medium and low zoom levels, an additional geometric figure to be used in combination with squares must be individuated. Such geometric

figure has to be detectable even in presence of blurred images.

Taking into account the above considerations, we developed a novel calibration pattern that is shown in Fig. 3a. It is a  $11 \times 11$  chessboard containing 454 circles inside. Since the pattern is made of two different types of features (i.e., squares and circles), it is possible to consider such features together or separately, depending on the zoom level, to calibrate the camera.

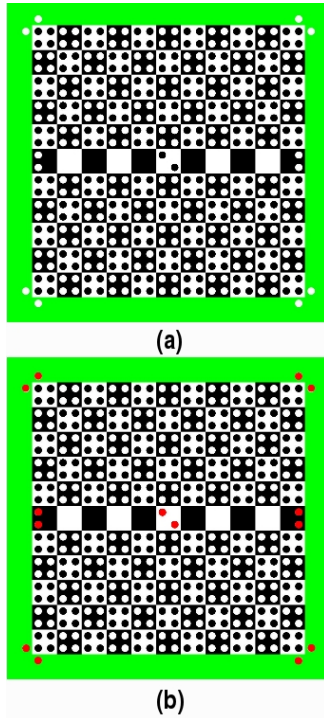


Figure 3: a) The novel calibration pattern. b) The 14 reference points highlighted in red.

The pattern has a coloured (green) border that is useful for individuating it in the scene when low zoom levels are used. There are also 14 reference circles that can help in understanding which region of the pattern is currently viewed by the camera, in case the pattern is only partially captured (Fig. 3b).

### 3.2 Find Matrix

The Find Matrix module is responsible to detect the squares and the circles in the image and to find the correspondences between the found features and the model pattern (shown in Fig. 3a). The circle centres are individuated using an algorithm (see Alg. 1) based on the following OpenCV [Ope13a] functions: *threshold*, *findContours*, *approxPolyDP*, and *minEnclosingCircle*.

Given the captured frame, a binary image *BImg* is obtained applying a thresholding process, then the above listed functions are used to find the contours in *BImg*, the approximation of the contours to the best fitting

---

**Input:**

*binary image BImg*;

**Output:**

*set of pairs (center, radius) < ctrs, r >*;

**Data Structures:**

*set of contours cont*;

*set of closed contours contPol*;

**Initialize:**

$\forall_i \text{ ctrs}[i] = 0, \text{cont}[i] = 0, \text{contPol}[i] = 0, r[i] = 0$ ;

$\text{cont} \leftarrow \text{findContours}(BImg)$ ;

**for** *cont* [*i*] **do**

$\text{contPol} \leftarrow \text{approxPolyDP}(\text{cont}, \epsilon, \text{closed})$ ;

$\text{ctrs}, r \leftarrow \text{minEnclosingCircle}(\text{contPol})$ ;

$\text{ctrs}, r \leftarrow \text{RefineCenters}(\text{ctrs}, r)$ ;

---

Algorithm 1: Find Circle Centers

polygons, and the approximation of those polygons to circles. Notice that the pattern does not need to be perfectly aligned or rectified in the image, since corner and circle detection are robust enough to variations of orientation of the pattern.

In order to eliminate the false positive detections, the list of centres is refined by eliminating the polygons that are too large or too small according to a parametric threshold  $T$ , computed as:

$$T = \frac{\sum_{i=1}^n \text{radius}_i}{n} \quad (1)$$

where  $\text{radius}_i$  is the  $i$ -th radius belonging to the set  $S$  of detected circles and  $n$  is the cardinality of  $S$ .

We observed that when the zoom level is low, i.e., the captured image of the pattern is small with respect to the camera frame, many false positives arises in detecting the circles due to the difficulty of extracting well defined shapes. However, for high zoom levels, even if the image of the pattern can result blurred, our algorithm is able to find all the circles in the visible portion of the pattern as well as all the circle centres (Fig. 4).

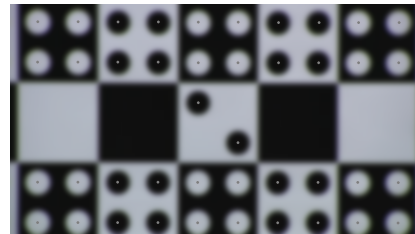


Figure 4: The *Find Circle Centres* algorithm is robust even in presence of a blurred image as input.

For doing this, the *Find Matrix* Algorithm (Alg. 2) takes as input the dimensions  $w$  and  $h$  of the circle grid in the model pattern, a point  $p$  (namely, the point that has the smaller abscissa among all the found points in the pattern), four threshold values  $T$ ,  $offset$ ,  $\delta$ ,  $\epsilon$

**Input:**  $\langle ctrs, r \rangle, T, offset, p$

**Output:**  $M$

**Local Variables:**

$thresholds \delta, \epsilon$

**Data Structures:**

set of current row points  $rpnts$

binary matrix  $M$

average of current set of radii  $av$

**Initalize:**

$\forall_{i,j} M(i,j) = 0, \forall_k rpnts[k] = 0$

**for each**  $ctrs[i]$  **do**

$\langle rpnts \rangle \leftarrow findRow(ctrs, i)$

$av \leftarrow computeAverage(rpnts)$

**for each**  $rpnts[j]$  **do**

$\delta \leftarrow av + p + (j * T) - offset$

$\epsilon \leftarrow av + p + (j * T) + offset$

**if**  $rpnts[j] < \epsilon, rpnts[j] > \delta$  **then**

$M(l, m) = 1;$

**else**

$M(l, m) = 0$

$deleteCenters(rpnts, \langle ctrs, r \rangle)$

$M \leftarrow CropMatrix(M)$

Algorithm 2: Find Matrix

that are related to the dimensions of the pattern and can be experimentally found, and a set of pairs  $\langle ctrs, r \rangle$  (i.e., the list of the found centres together with their relative radii).

The procedure starts by initialising with zeros the  $w \times h$  matrix  $M$ . Then, the algorithm takes the first element in the list  $ctrs$  and searches for points that have the ordinate value included in a range defined by the thresholds  $\delta$  and  $\epsilon$ . All the points that satisfy such constraints form a *circle row*. The obtained row is sorted according to the abscissa values of its points and it is compared to the corresponding row model of the known pattern. A vector  $rpnts$  is filled with values “1” if the point is included in the row model, “0” otherwise. By iterating the above steps, a matrix  $M$  composed by “1” and “0” is obtained. Finally, if  $M$  is smaller than the model pattern, a cropping procedure is applied to reduce its dimensions.

Once  $M$  has been computed, a graph matching between  $M$  and a matrix  $C$ , that represents the model pattern, can be applied. It is worth noting that each image used for the calibration procedure may contain at least a reference point in it. This is a reasonable assumption, since the reference points are distributed along the whole pattern (see Fig. 3b). Thus, we assume that the captured portion of the pattern is composed by a set of circles and by at least one group of reference points. Since the patterns of the reference points are all different each other, the observation of a single one is sufficient to determine the portion of the calibration pattern that is captured.

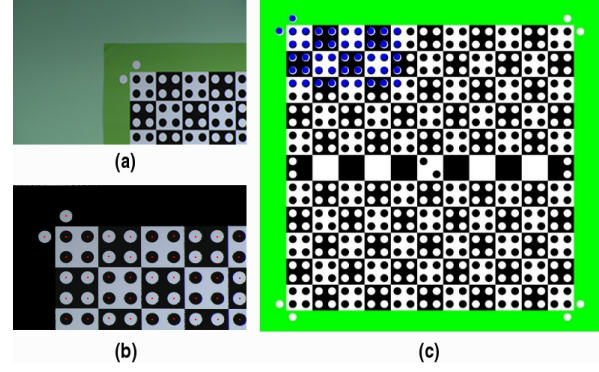


Figure 5: Recognition example for a partially captured pattern. a) The image in input. b) The detected centers. c) The graph matching procedure correctly assigns the observed centres to the corresponding ones in the model pattern.

As an example, if the current image of the pattern is the one shown in Fig. 5a, the matrix  $M$  will be:

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Given the matrix  $C$  that represents the model pattern:

$$C = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 & \dots & 0 & 1 & 0 \\ 1 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 1 \\ 0 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\ 0 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\ 0 & 1 & 0 & \dots & 1 & 0 & \dots & 0 & 1 & 0 \\ 0 & 1 & 0 & \dots & 0 & 1 & \dots & 0 & 1 & 0 \\ 0 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\ 0 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\ 1 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 1 \\ 0 & 1 & 0 & \dots & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}$$

it is possible to find the portion of  $C$  corresponding to  $M$  (see Fig. 5c).

Fig. 6 shows the in-lab pattern that we used. It is the combination of two above discussed model patterns (one with red and one with green borders). The dimensions for each one of the two parts of the in-lab pattern are  $2m \times 1m$  with 18 rows and 34 columns of circles, each circle having a radius of 1.5 cm and being at a distance of 5 cm (center-center) from the others. The two parts of the in-lab pattern are positioned with an angle of 90 degrees in order to have non-coplanar points. The two different border colours facilitate the detection of the in-lab pattern in the scene.



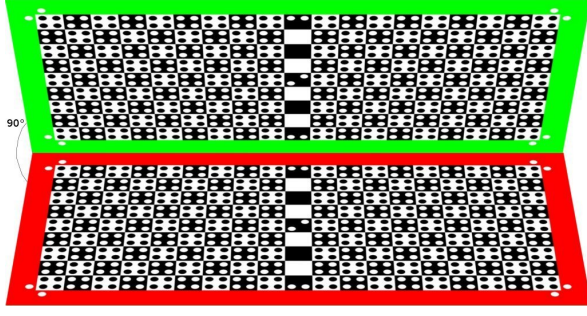


Figure 6: In-lab pattern.

### 3.3 Intrinsic Parameter Recovery

A series of images containing the in-lab pattern at different zoom levels are collected to determine the intrinsic parameters. Once the key points are obtained using the Find Matrix module, a common technique [Sir04a] is applied to calculate five intrinsic parameters:  $f$ : focal length;  $(s_x, s_y)$ : pixel size in  $x, y$ ;  $(u, v)$ : principal point. Usually, it is possible to assume square pixels and so  $s_x = s_y = s$ . The intrinsic matrix  $K$  can be defined as follows:

$$K = \begin{bmatrix} f_x & s & u_0 \\ 1 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where  $s$  is the skew parameter that represents the angle between the  $x$  and  $y$  pixel axes,  $f_x = f/s_x$ ,  $f_y = f/s_y$  are the focal lengths.

The quantitative evaluation for the in-lab calibration will be given in Section 5.

### 3.4 On-field Pattern

The on-field step of the proposed method requires the use of a second calibration pattern (Fig. 7a), made of three coloured rectangles as background, 19 black circles, and a white “+” sign drawn in the central black circle (Fig. 7b). We realized an on-field pattern of 2 m width and 1 m height.

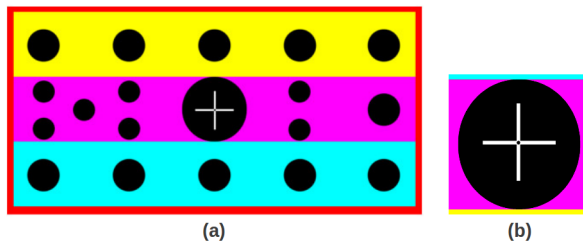


Figure 7: a) On-field pattern. b) A detail of the “+” sign in the center of the pattern.

In order to perform the extrinsic parameter estimation of the camera, two short videos have to be collected on the field:

1. The pattern is captured from five different angles of views with a fixed zoom level (Fig. 8);
2. The center of the pattern is captured starting from the maximum zoom level until reaching the minimum level (Fig. 9).



Figure 8: The on-field pattern is captured from 5 different angles of view.

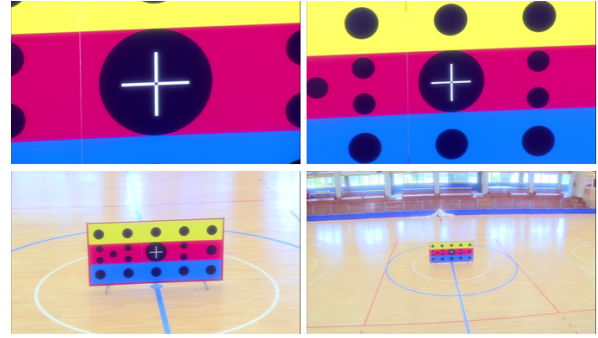


Figure 9: The “+” sign in the center of the on-field pattern is captured at different levels of zoom.

Each video is about 2 minutes long. The aim of the first video is to obtain a set of image points positioned in the center and in the four corners of the image. The second video is used to track the “+” sign over different levels of zoom. The extrinsic parameter calculation using the two videos is detailed in the following.

### 3.5 Extrinsic Parameter Estimation

Extrinsic parameters play a crucial role in the camera calibration process, since those parameters define the location and orientation of the camera with respect to the world reference frame.

To obtain such parameters the algorithm proposed by Tsai in [Tsa86a] and the calibration method by Davis and Chen [Dav03a] are used.

Let  $(u, v)$  be the ideal (distortion-free) pixel image coordinates, and  $(\tilde{u}, \tilde{v})$  the corresponding real observed image coordinates. The ideal points are the projection of the model points according to the pinhole model. Similarly,  $(x, y)$  and  $(\tilde{x}, \tilde{y})$  are the ideal (distortion-free) and real (distorted) normalized image coordinates. As reported in [Zhang00a] we have:



$$\begin{aligned}\tilde{x} &= x + x[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \\ \tilde{y} &= y + y[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]\end{aligned}$$

where  $k_1$  and  $k_2$  are the coefficients of the radial distortion.

The function for the radial distortion is here approximated through the first two terms of the development of the series  $f(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots$ , in such a way that only the parameters  $k_1$  and  $k_2$  are considered.

### 3.6 Parameter Regularization

In order to regularize the intrinsic and extrinsic parameter functions (depending on zoom levels), several methods can be applied. Since no information about the physics of the optical phenomena have been used, a "black box" approach is suitable. We compared polynomial fitting versus Artificial Neural Network (ANN) [Fan13a] and chose the latter because it provides better results.

The second video collected during the on-field calibration phase, that contains the "+" sign captured at different zoom levels, is used to refine the calibration parameters, in particular, the principal point  $(u, v)$  and the focal lengths  $f_x$  and  $f_y$ , and the radial distortion coefficients  $k_1$  and  $k_2$  are considered.

Two different ANNs have been implemented, the former for managing the lower zoom levels, the latter for the higher ones. Indeed, from Fig. 10 it is noticeable that the trends for  $f_y$  (Fig. 10a) and for the first coefficient of radial distortion  $k_1$  (Fig. 10b) are quasi-linear for low zoom levels, while they become non-linear for higher zoom levels.

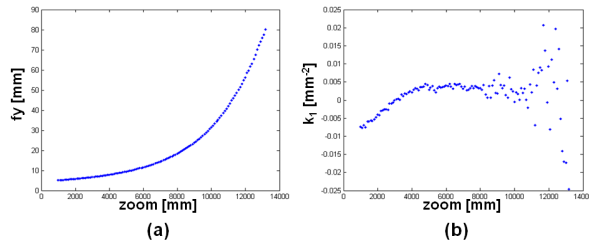


Figure 10: Variation of camera parameters in function of zoom. a) Focal length along y axis. b) First coefficient of radial distortion.

The neural network for the lower zoom levels includes a hidden layer made of two units, while the neural network for the higher levels has a hidden layer composed by six units. This choice has been taken for allowing the ANN that manages the non-linear zone to have a greater number of degrees of freedom, while inserting more units in the hidden layer of the ANN that manages the quasi-linear zone would have generated overfitting problems. Two results of the regularization process are shown in Fig. 11.

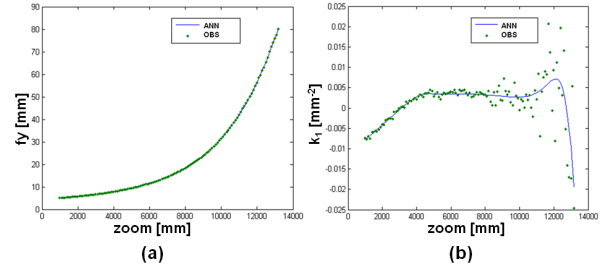


Figure 11: Real observations (OBS) are reported as green dots, while the functions find by the two ANNs are reported as blue lines. a) Focal length along y axis. b) First coefficient of radial distortion.

In order to allow a smooth transition between the two regularization functions calculated by the ANNs, a third (overlapping) zone, where a regularization function is computed as a linear combination of the other two functions, is considered (see Fig. 12). Let  $f_1$  be the return value of the first neural network computed on a value  $x$  and  $f_2$  be the function generated by the second neural network for the same value  $x$ . Let  $I_0$  and  $I_1$  be the two extremities of the central green stripe in Fig. 12, we have:

$$\lambda = \frac{x - I_0}{I_1 - I_0};$$

and  $f_3(x)$  will be:

$$f_3(x) = (1 - \lambda)f_1(x) + \lambda f_2(x)$$

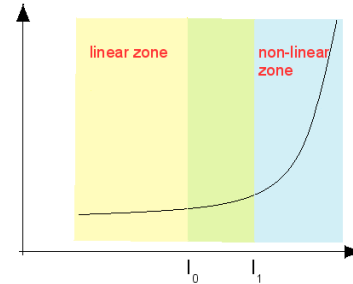


Figure 12: Overlapping zone where a regularization function is computed as a linear combination of the two functions computed by the ANNs.

## 4 APPLICATION SCENARIO

Virtual advertising concerns the use of computer vision techniques to insert virtual advertising images into live or pre-recorded television shows (e.g., sport events). The aim is to put virtual advertising billboards<sup>2</sup> on the play field (Fig. 13).

<sup>2</sup> The trade marks shown in Figs. 13 and 14 have been randomly chosen for demonstrating the output of the system and are not involved in the work described in this paper.

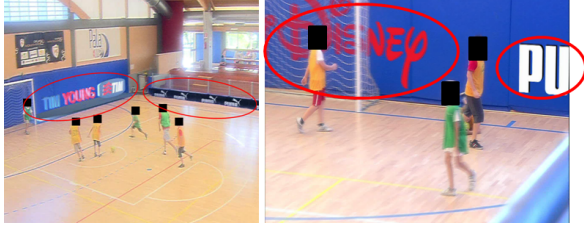


Figure 13: Virtual advertising example. Virtual billboards (highlighted by the red ellipses) are positioned in the scene using a chroma-keying technique.

This particular application scenario is a challenging one, since a lot of changes in the zoom levels are required for adequately capturing sport events. Thus, an accurate camera calibration is needed for the replacing process, since the projection of the billboard to be replaced onto the image plane must be known. Moreover, the calibration model must be robust to camera zoom variations, since the replacement must be dynamic to adapt to the varying zooming parameters (Fig. 14).

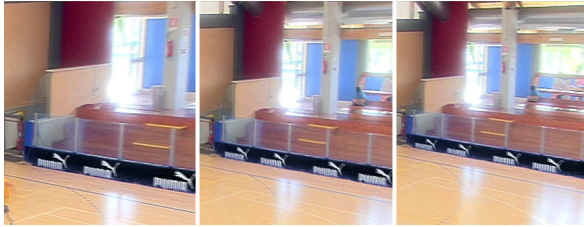


Figure 14: The proposed approach is robust to zoom variations. This figure shows three different screen shots resulting from three different zoom values.

## 5 EXPERIMENTAL RESULTS

A quantitative evaluation has been carried to demonstrate the effectiveness of the approach. For measuring the accuracy of the in-lab calibration, a set of views of the in-lab pattern captured with a professional HD camera ( $1920 \times 1080$  resolution) has been collected. The accuracy of the intrinsic parameters has been calculated at different zoom levels and with different angles of view.

The results generated by using our novel calibration pattern are compared with those obtained by using a traditional pattern. The comparison has been performed by manually determining a ground truth that has been compared with the re-projection values generated by the two methods. The quantitative results of the evaluation are reported in Table 1 and the error is measured in terms of pixels. For each considered zoom level 20 different images have been examined.

The error values are also plotted in Fig. 15 to highlight that the proposed method generates a constant error (about 0.5 pixel), while the traditional procedure introduces an higher error with a more variable trend.

Zoom Level	Trad. Calib. Error		Prop. Calib. Error	
	Avg.	Std. Dev.	Avg.	Std. Dev.
1000	0.68	0.32	<b>0.56</b>	0.26
1500	0.70	0.34	<b>0.52</b>	0.25
1800	0.75	0.36	<b>0.53</b>	0.25
2000	0.70	0.33	<b>0.51</b>	0.24
2400	0.72	0.35	<b>0.53</b>	0.26

Table 1: Quantitative comparison of the re-projection error obtained by using a traditional pattern based calibration and by adopting the proposed method.

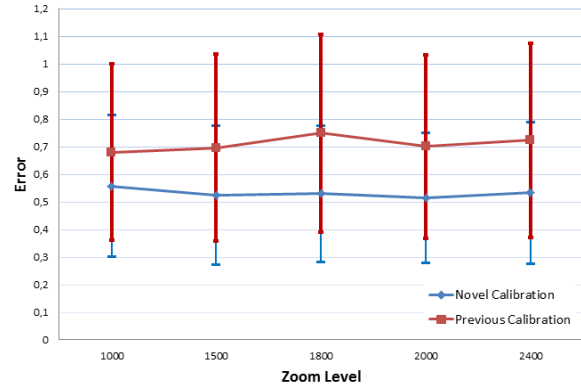


Figure 15: The data plotted represent the re-projection error made by using a traditional pattern (red line) and by adopting the proposed novel calibration pattern (blue line). The standard deviation is reported as error bars.

We evaluated also the time required for both the novel and the previous (used by the company) calibration procedures (see Fig. 1). The results described in Table 2 report the time difference of the calibration procedures in the soccer environment shown in the figures of this paper. Calibration time has been measured considering a within-subject design factor (the same human operator performed both the calibration approaches in the same operative scenario). The test has been carried out by an expert employee of the company, familiar with the previous method and properly instructed to use the novel one.

Multiple calibration runs have not been performed, because they would have required a different scenario, since making calibration with the same data multiple times would not provide relevant statistical evidence. Nonetheless, the difference in time is significant and the expert employee confirmed that other previous calibrations were in the same order of magnitude of the one reported in this table.

## 6 CONCLUSIONS

In this paper a novel approach for calibrating zooming cameras is presented. The method uses two novel calibration patterns, the former composed by squares and circles and designed for recovering the intrinsic parameters of the camera even for high zoom levels, the latter conceived for calculating the extrinsic parameters of

Time	Previous Method	Novel Method
In-lab calib.	2880 min.	30 min.
On-field calib.	90 min.	10 min.
Billboard Identification	10 min./bill.	0.5 min./bill.

Table 2: Calibration time using the previous calibration procedure and the novel one. The test has been carried out by an expert user.

the camera and for making the calibration process easier and faster.

A quantitative evaluation demonstrates: 1) The increase of accuracy of the proposed method for the intrinsic parameter estimation with respect to a traditional calibration pattern; 2) The decrease of the calibration time for the on-field calculation of the extrinsic parameters of the camera.

A practical application of the method for creating a virtual advertisement system has been described. The system is currently used by a company providing real time virtual advertisement services for sport events. The chosen application scenario is a challenging one, since a lot of changes in the zoom levels are required for adequately capturing sport events.

As future work we intend to analyse the problem of automatically individuating possible occlusion in front of the billboards that will be replaced.

## ACKNOWLEDGEMENTS

This work has been realized within the project FILAS-RS-2009-1286 “Advanced 3D virtual advertisement system (Pixidis)” supported by Duel TV S.p.A. ([www.dueltv.com](http://www.dueltv.com)) and FILAS Lazio ([www.filas.it](http://www.filas.it)). The authors thank in particular Riccardo Colasanti, Antonio Di Noto, and Edoardo Egidi from Duel TV for their continuous support and for providing means and technical expertise to carry out this research.

## 7 REFERENCES

- [Aga01a] Agapito, L., Hayman, E., and Reid, I. Self-calibration of rotating and zooming cameras. *Int. J. Comput. Vision*, vol. 45, pp. 107-127, 2001.
- [AlA06a] Al-Ajlouni, S., and Fraser, C. Zoom dependent camera calibration. *American Society of Photogrammetry and Remote Sensing*, 2006.
- [Dav03a] Davis, J., and Chen, X. Calibrating Pan-Tilt Cameras in Wide-Area Surveillance Networks. *IEEE Int. Conf. on Computer Vision*, pp. 144-150, 2003.
- [Fan13a] FANN - Fast Artificial Neural Networks. <http://leenissen.dk/fann/wp/>
- [Har04a] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.
- [Kim00a] Kim, H., and Hong K. S. A Practical Self-Calibration Method of Rotating and Zooming Cameras. *Int. Conf. on Pattern Recognition*, vol. 1, pp.1354, 2000.
- [May97a] Maybank, S. J., and Faugeras, O. A theory of self calibration of a moving camera. *Int. J. of Comp. Vis.*, vol. 8, pp. 123-151, 1992.
- [Oh09a] Oh, J., Nam, S., and Sohn, K. Practical Pan-Tilt-Zoom-Focus Camera Calibration for Augmented Reality. *Int. Conf. on Computer Vision Systems*, pp. 225-234, 2009.
- [Ope13a] OpenCV (Open Source Computer Vision), <http://opencv.org>
- [Sal02a] Salvi, J., and Armangué, X., and Batlle, J. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, vol. 35, no. 7, pp. 1617-1635, 2002.
- [Sarkis07a] Sarkis, M., Senft, C. T., and Diepold, K. Modeling the Variation of the Intrinsic Parameters of an Automatic Zoom Camera System using Moving Least-Squares. *Int. Conf. on Automation Science and Engineering*, pp. 560-565, 2007.
- [Sin06a] Sinha, S. N., and Pollefeys, M. Pan-tilt-zoom camera calibration and high-resolution mosaic generation. *Comput. Vis. Image Underst.*, vol. 103, n. 3, pp. 170-183, 2006.
- [Sir04a] Sirisantamrid, K., Matsuura, T., and Tirasesth, K. A simple technique to determine calibration parameters for coplanar camera calibration. *TENCON*, vol. 1, pp. 677-680, 2004.
- [Stu97a] Sturm, P. Self-calibration of a moving zoom-lens camera by pre-calibration. *Image and Vision-Computing*, vol. 15, pp. 583-589, 1997.
- [Tsa86a] Tsai, R. Y. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 364-374, 1986.
- [Zhang00a] Zhang Z. A Flexible New Technique for Camera Calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330-1334, 2000.



# Constrained Up-Scaling for Direct and Global Image Components

Julian Bader

Martin Pätzold

Andreas Kolb

Computer Graphics Group, Institute for Vision and Graphics, University of Siegen  
 Hölderlinstraße 3, 57076 Siegen, Germany  
 { julian.bader,martin.paetzold,andreas.kolb }@uni-siegen.de

## Abstract

The separation of direct and global illumination components is interesting for many applications in Computer Graphics and Computer Vision, such as BRDF estimation or material classification. However, for full-resolution images, a large number of coded images have to be acquired. For many interactive applications, such as the acquisition of dynamic scenes or video capturing, this is not feasible. In this paper, a new constrained up-scaling technique for separated direct and global illumination images is proposed which requires two to three coded input images, only. Our approach imposes the boundary condition that the sum of the direct and global components equals the fully illuminated image. We work in a predictive-corrective manner where we first use a single-image up-scaling method in order to predict the higher resolution images. Afterwards, the missing higher frequencies are determined using a fully illuminated image. As the distribution of the higher frequencies differs among the various frequency bands, we apply our approach in an iterative way for small up-scaling steps distributing the missing information by minimizing the overall frequencies. We evaluate the up-scaling scheme and demonstrate the improvement compared to single-image approach. As our method aims at minimizing the structured light patterns needed for acquisition, we additionally discuss the performance of existing pattern sets in terms of applicability for dynamic scenes.

## Keywords

Computational Photography, Image Processing, Global Illumination

## 1 INTRODUCTION

The separation of direct and indirect illumination in a scene is an interesting task for both, Computer Graphics and Computer Vision domains. The direct component helps obtaining the distribution of reflected light in a scene. This can be used to derive a model for the bidirectional reflection distribution function (BRDF) [GKGN11]. Also, 3D reconstruction methods are more robust when working only on the direct component [GKGN11, NG12]. The global component gives insight into the scattering behaviour between one or more objects in a scene. This is useful to achieve more photo-realistic renderings of a scene by observing the complex light flow [MYR10]. Also, object recognition benefits from such approaches as the scattering of an object highly depends on the material [GL12]. In 2006, Nayar et al. [NKGR06] presented an approach for separating the direct and global illumination using a structured-

light approach. However, for a high resolution separation, a large number of structured-light patterns is needed. Thus, this approach is not feasible for dynamic scenes, such as capturing biometric information of human faces or mobile navigation, or for video capturing. In their work, an alternative method was proposed where only one structured light pattern is sufficient. The disadvantage of this method is that the image resolution decreases dramatically. As a consequence, small scale details of fine textured regions are lost.

In this paper, we propose a new constrained image up-scaling technique for direct and global component images. We employ an iterative prediction-correction approach which uses a single-image up-scaling method in order to predict the higher resolution representation for each component. These up-scaled images are then corrected according to a high resolution fully illuminated image which has been additionally acquired. Fine structures in images, such as textiles or fur, can be recovered although they are not apparent in the low resolution input image. Our contributions are the following:

- A general constrained up-scaling scheme for the transfer of high frequency information.
- The application of the proposed scheme for direct and global component images.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

- An evaluation of various low-resolution separation approaches which shows that the resulting image quality is not sufficient for the proposed up-scaling scheme.

To the best of our knowledge, we are the first proposing a constrained up-scaling approach for direct and global component images.

The rest of the paper is structured in the following way: Sec. 2 gives an overview of existing single-image and multi-modal up-scaling techniques. In Sec. 3, the principles of the separation approach of [NKGR06] is reviewed and the impact of the different structured light patterns is discussed. Our constrained up-scaling method is described in Sec. 4. In Sec. 5, the results are discussed. Furthermore, the improvement of our approach compared to the single-image prediction is shown. Finally, Sec. 6 concludes the paper.

## 2 RELATED WORK

In the past, the problem of image up-scaling has been addressed many times during the past 20 years. Here, a brief overview of the major approaches is given whereas we distinguish between the single-image and multi-modal approaches.

**Single-image methods.** The single-image up-scaling methods can be divided into four groups: region-based interpolation, edge-directed interpolation, edge-directed reconstruction and example-based reconstruction.

Region-based interpolation methods apply different interpolation algorithms depending on the region in the image. Thurnhofer and Mitra [TM96] proposed an image interpolation approach which classifies image pixels into three different regions (constant, irregular and oriented). Other methods distinguish only between two regions (oriented, homogeneous) for different interpolation scheme application [ABA01, CHL05]. Zi et al. [ZDLL12] proposed a perceptual motivated segmentation of regions. For the general and transition regions, different kinds of interpolation algorithms are used. The attention region is computed according to an energy formulation optimizing curvature continuity, curvature enhancement and isolevel curve smoothing.

Edge-directed interpolation schemes adjust the interpolation according to the edges present in the image. Li and Orchard [LO01] use the local covariance for an edge-directed interpolation. Alternatively, sharp luminance variations and discontinuities are considered [BGS02]. Su and Willis [SW04] proposed a pixel triangulation scheme in order to use only pixels for interpolation not separated by an edge. Other methods propose the use of the second order derivative in order to determine the interpolation direction [GA08, GA11] and iterative refinement.

Edge-directed reconstruction algorithms try to estimate a high-resolution edge image which is then used to reconstruct the up-scaled image. Tai et al. [TTT06] proposed to use tensor voting for high resolution edge reconstruction. In alternative approaches, edge statistics are learned in order to predict enhanced edge images [Fat07, SXS08, YXYC12]. Another kind of approach [YXXY12, WXM<sup>+</sup>13] computes sharp high resolution gradients directly from the low resolution images.

Example-based approaches try to build up a dictionary which relate low resolution image patches to missing higher frequencies [YWL<sup>+</sup>12]. Freeman et al. propose a nearest neighbour search in the training set to find the high frequencies for image patches [FJP02]. Kim and Kwon [KK08] formulated the relationship between low and high frequencies as a linear regression model in order to derive a suitable image patch. Alternatively, a single-hidden-layer feed-forward neural network is used for learning the relation between low and high images frequencies [AB12]. A variety of methods are based on self similarity [GADTH97, EV07, SSU08, GBI09, FF11]. Here, the assumption is that an image contains similar structures at different scale levels.

**Multi-modal methods.** In [LLK08] and [STDT08], methods are proposed for the combination of low resolution time-of-flight depths images with high resolution RGB images. However, instead of using the RGB information for a constrained up-scaling, both approaches first do a single-image up-scaling of the depth image before fusing with RGB data. Langmann et al. [LHL11, LHL12] describe a fusion approach of time-of-flight and RGB data where several cross bilateral filtering strategies are compared. Additionally, a framework for joint segmentation and super-resolution is proposed. They assume the smooth change of the depth field in homogeneous regions. Fine structures, such as in fur or textiles, can therefore not be recovered.

**Direct-Global Separation methods.** In 2006, Nayar et al. [NKGR06] proposed a method for separating the direct and global image components using high-frequency structured-light patterns. Gu et al. [GKGN11] extended this approach in order to use multiple light sources. In 2012, two approaches were presented which estimate the light transport in a scene [ORK12, RRC12]. Both are also capable of direct-global separation. However, O'Toole et al. use a complicated setup and long acquisition times. The method of Reddy et al. needs a high number of structured-light patterns. As we focus on the acquisition of dynamic scenes, the latter approaches are not suitable.

The aim of our approach is the reconstruction of small detail which get lost during single-coded-image separation process. As the above mentioned single-image up-scaling methods can handle singularities, such as



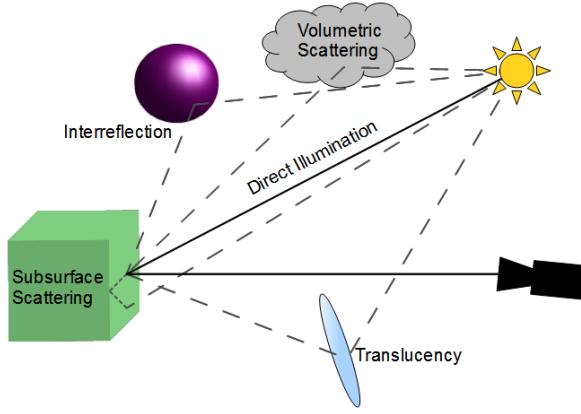


Figure 1: Light can reach the camera in different ways. Light being directly reflected to the camera is called *direct illumination*. *Global illumination* denotes light that is scattered in the scene and travels an indirect path to the camera.

edges or joints, quite well, fine homogeneous structures, such as fur or threads in textiles, cannot be reconstructed. Due to the smoothness assumption of depth data, the multi-modal super-resolution methods for fusion of time-of-flight data also cannot cover these kind of fine details. In contrast, our proposed method uses a single-image up-scaling prediction for the direct and global component images and enforces the sum of these two to be equal to a fully illuminated image.

### 3 SEPARATION OF DIRECT AND GLOBAL ILLUMINATION

The illumination in a scene can be decomposed into two parts. The first part is the so called direct illumination. This covers all the light which is emitted by a light source and is reflected directly to the scene observer (usually a camera or the human eye). The second part addresses all the light which is first scattered or otherwise redirected and reaches the observer in an indirect path. This is considered as global illumination (cf. Fig. 1).

#### 3.1 Separation Scheme

In 2006, Nayar et. al [NKGR06] proposed a method to separate the direct and global illumination in a scene using structured light. Here, the underlying assumption is that the scattering of light, which is responsible for the global illumination effects, acts as a kind of low-pass filter, i.e. there are no high frequencies present in the global component. Thus, by projecting different high frequency patterns onto the scene, the following relations can be formulated depending on whether a part of the scene is hit by a light or dark source element:

$$L^+(\mathbf{i}) = L_d(\mathbf{i}) + \alpha L_g(\mathbf{i}) + b(1 - \alpha)L_g(\mathbf{i}), \quad (1)$$

$$L^-(\mathbf{i}) = bL_d(\mathbf{i}) + (1 - \alpha)L_g(\mathbf{i}) + \alpha bL_g(\mathbf{i}). \quad (2)$$

Here  $L^+(\mathbf{i})$  refers to the intensity value of a scene element at position  $\mathbf{i} = (x, y)^T$  which is directly illuminated, whereas  $L^-(\mathbf{i})$  represent the intensity when the element is not directly illuminated.  $L_d$  and  $L_g$  refer to the direct and global part of the illumination and  $\alpha$  is the fraction of projector source elements which are lit. Since digital projectors cannot be tuned completely dark,  $b$  accounts for the fraction of a white source element emitted by a "black" projected pixel with  $0 \leq b \leq 1$ . For a more detailed description of the separation method, the reader is referred to the paper of Nayar et al. [NKGR06].

#### 3.2 Structured-Light Patterns

Nayar et al. [NKGR06] proposed several high-frequency structured-light patterns in order to acquire  $L^+(\mathbf{i})$  and  $L^-(\mathbf{i})$  for a scene.

**Checker-board.** The scene is illuminated by a sequence of shifted checker-board patterns. For each pixel in a scene the brightest and the darkest intensity value among all the acquisitions is chosen. With this kind of patterns, one can compute the direct and global component directly at the camera's resolution. However, due to the acquisition of multiple source images, this method is not feasible for the fast acquisition of scenes.

**Phase shifted sinusoidal.** Each pixel of the pattern can be expressed as  $p_0 = 0.5(1 + \sin \phi)$  for  $\phi \in [0, 2\pi]$ . Analogue to the first pattern, two more are generated where  $\phi$  is phase shifted by  $\frac{2\pi}{3}$  and  $\frac{4\pi}{3}$ . In general, the pixels of the first pattern can be chosen arbitrarily, but in order to ensure high frequency, a sinusoidal function which varies in both, x- and y-direction, should be chosen. Using these patterns, only three acquisitions are necessary. However, Gu et al. [GKGN11] stated that this method suffers from serious image artefacts. Therefore, it is not suitable if the capturing of fine details is required.

**Stripes.** By using a horizontal or vertical stripe pattern, only one acquisition is necessary. Here, the brightest and darkest pixels are determined in small rectangular patches which are aligned perpendicular to the stripes. These extrema are interpolated and averaged in order to generate the minimum  $L^+(\mathbf{i})$  and maximum  $L^-(\mathbf{i})$ . Then, the direct and global components are computed. As only one pattern needs to be acquired, this method is suitable for capturing dynamic scenes. However, due to the averaging process, the resolution of the resulting images is decreased by a fraction of four [NKGR06].

### 4 A CONSTRAINED UP-SCALING METHOD

The aim of our approach is the constrained up-scaling of direct and local component images according to one

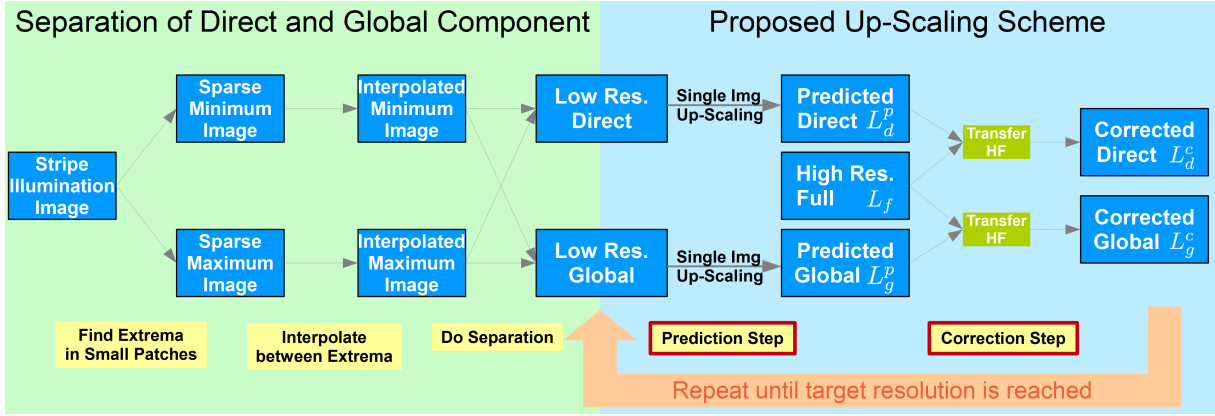


Figure 2: Overview of the complete system. The left side (green) shows the separation of global and direct components according to [NKGR06]. The right side (blue) describes our constrained up-scaling technique.

high-resolution fully illuminated image whereas we focus on the reconstruction on fine structures in homogeneous regions. Before the details are discussed in the following two sections, a rough system overview is given.

As input images, we take the two low resolution global and direct component images retrieved by the stripe pattern separation approach proposed by Nayar et al. [NKGR06]. Additionally we use a high resolution image which is acquired with full illumination. The proposed up-scaling scheme consists of two steps which are applied iteratively until we reach the target resolution (cf. Fig. 2):

1. In the **prediction** step, we estimate the higher resolution image using a single-image up-scaling technique. The first iteration takes the low-resolution direct and global component images as input. For all following iterations, the corrected images from the previous iteration are used.
2. The **correction** step corrects the predicted images according to the main condition that the sum of direct and global images results in the fully illuminated image. In order to decrease the noise level in the up-scaled image, the error to the full image is distributed in such a way that the overall intensity of higher frequencies is minimized.

#### 4.1 Prediction

In general, there are no limitations regarding the prediction step in our up-scaling scheme. As the correction works on the predicted image directly, every single-image up-scaling method could be used. In our experiments, we used the so-called Local Self-Examples approach proposed by Freedman and Fattal [FF11]. This method has two advantages. On the one hand, it can predict the object boundaries quite well. This improves the performance of the correction step, as there are lower intensity errors. On the other hand, the approach

increases the resolution of the input images iteratively by nondyadic scales (eg. 5:4 or 3:2). Therefore, in each iteration only a small higher frequency band is added to the image, leading to a more robust correction in terms of frequency distribution.

#### 4.2 Correction

After the prediction step, the image is corrected according to the constraint

$$L_f(\mathbf{i}) = L_d^p(\mathbf{i}) + L_g^p(\mathbf{i}) \quad (3)$$

where  $L_f(\mathbf{i})$  is the pixel at position  $\mathbf{i}$  under full illumination.  $L_d^p(\mathbf{i})$  and  $L_g^p(\mathbf{i})$  correspond to the (unconstrained) predicted direct and global pixels respectively. Taking the difference of the sum of the direct and global images and the fully illuminated image

$$D(\mathbf{i}) = L_f(\mathbf{i}) - (L_d^p(\mathbf{i}) + L_g^p(\mathbf{i})), \quad (4)$$

we get the error introduced by the prediction step. Please note that, similar to the *Difference of Gaussians*,  $D(\mathbf{i})$  only contains the missing higher frequencies which could not be recovered by the single-image up-scaling approach. In order to correct the predicted images, we distribute the error via two functions  $f_d(\mathbf{i}, L_d^p, D)$ ,  $f_g(\mathbf{i}, L_g^p, D)$  so that

$$D_{\mathbf{i}} = (f_d(\mathbf{i}, L_d^p, D) + L_d^p(\mathbf{i})) + (f_g(\mathbf{i}, L_g^p, D) + L_g^p(\mathbf{i})) \quad (5)$$

holds. These functions should be designed in such a way that they select these higher frequencies which are missing in the corresponding predicted image. If we now assume  $f_d$  and  $f_g$  to be linear with respect to  $D$ , we express our corrected images as follows

$$L_d^c(\mathbf{i}) = L_d^p(\mathbf{i}) + \alpha_d(\mathbf{i}) \cdot D(\mathbf{i}), \quad (6)$$

$$L_g^c(\mathbf{i}) = L_g^p(\mathbf{i}) + \alpha_g(\mathbf{i}) \cdot D(\mathbf{i}) \quad (7)$$

where  $L_d^c$  and  $L_g^c$  are the corrected direct and global component images, respectively. Here,  $\alpha_d$ ,  $\alpha_g$  represent the weights of distributing the prediction error  $D$




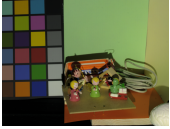

										
	Scene 1		Scene 2		Scene 3		Scene 4		Scene 5	
	corr.	uncorr.	corr.	uncorr.	corr.	uncorr.	corr.	uncorr.	corr.	uncorr.
RMSE (d)	0.0067	0.0078	0.0061	0.0113	0.0088	0.0194	0.0057	0.0101	0.0090	0.0158
RMSE (g)	0.0063	0.0073	0.0057	0.0081	0.0085	0.0115	0.0058	0.0084	0.0088	0.0114
$\ell_1$ -Norm (d)	11370	14504	12167	20909	16738	36631	9788	16989	16639	30017
$\ell_1$ -Norm (g)	9503	12133	11334	15698	15399	20194	9633	13476	15084	18611
$\ell_2$ -Norm (d)	12.79	15.03	11.23	20.77	16.18	35.63	10.49	18.54	16.49	28.95
$\ell_2$ -Norm (g)	12.02	14.08	10.51	14.89	15.70	21.11	10.66	15.35	16.16	20.99

Table 1: Comparison of five test scenes with different error metrics (root mean square error (RMSE),  $\ell_1$ - and  $\ell_2$ -norm): Low resolution images with separated direct(d) and global(g) illumination were scaled up by a factor of 2 with the approach of [FF11]. The values of the corrected image can be compared with the uncorrected image for each scene.

to both images on a per-pixel level. Now, we need a condition how to set the  $\alpha$ -values. As the wrong distribution tends to increase the noise level, a reasonable condition is to choose the weights in such a way that the present high frequencies are minimized. Thus, we can formulate a minimization problem where the sum of the squared Laplacian ( $\Delta$ ) of both images is minimized on a per-pixel level:

$$(\alpha_d(\mathbf{i}), \alpha_g(\mathbf{i})) = \arg \min_{\alpha_d, \alpha_g} \{(\Delta L_d)^2 + (\Delta L_g)^2\}. \quad (8)$$

Furthermore, we satisfy the constraints  $0 \leq \alpha_d, \alpha_g \leq 1$  and  $\alpha_d + \alpha_g = 1$  in order to eliminate the prediction error  $D$ . Solving for  $\frac{\partial F}{\partial \alpha_d} = 0$  and  $\frac{\partial F}{\partial \alpha_g} = 0$  with  $F = (\Delta L_d)^2 + (\Delta L_g)^2$  yields

$$\alpha_d(\mathbf{i}) = \frac{\Delta L_d^p(\mathbf{i})}{\Delta D(\mathbf{i})}, \alpha_g(\mathbf{i}) = \frac{\Delta L_g^p(\mathbf{i})}{\Delta D(\mathbf{i})}. \quad (9)$$

The constraints can easily be satisfied by normalizing each  $\alpha_d, \alpha_g$  with the sum  $\alpha_d + \alpha_g$ . Thus, the resulting correction terms for direct and global component images can be expressed as

$$L_d^c(\mathbf{i}) = \frac{\Delta L_d^p(\mathbf{i})}{\Delta L_d^p(\mathbf{i}) + \Delta L_g^p(\mathbf{i})} \cdot D(\mathbf{i}) + L_d^p(\mathbf{i}), \quad (10)$$

$$L_g^c(\mathbf{i}) = \frac{\Delta L_g^p(\mathbf{i})}{\Delta L_d^p(\mathbf{i}) + \Delta L_g^p(\mathbf{i})} \cdot D(\mathbf{i}) + L_g^p(\mathbf{i}). \quad (11)$$

The direct and global component images  $L_d^c, L_g^c$  are then given as input for the prediction step of the next iteration until the target resolution is reached.

## 5 RESULTS

In order to evaluate our up-scaling scheme and the low-resolution separation methods, a measurement setup consisting of a projector (Panasonic PT-LC76E) and

a single-lens reflex camera (Nikon D300) was used. Structured-Light patterns are projected into the scene that is imaged by the camera. The direct and global illumination for the given scene are separated by the approach of Nayar et al. [NKGR06]. In order to get a reliable ground truth, 25 shifted checker-board patterns were used.

In Sec. 5.1, we evaluate our up-scaling scheme. Therefore, we sampled down the ground truth and used these low-resolution images as input. Five test scenes were created for evaluating the influences of different scene properties, the impact of our correction for several predictors and the influence of the up-scaling factor. Sec. 5.2 shows how our up-scaling scheme performs on low-resolution direct and global component images produced directly using different structured-light patterns. We focussed on patterns which require up to three acquisitions, only.

### 5.1 Evaluation of the Up-Scaling Scheme

Tab. 1 shows different error values for the test scenes and the separated illumination. The error of the corrected image can be compared with the error of the uncorrected image for each test case. The error values in scene 1 look similar while the error values in scene 3 are reduced by half. Depending on the properties of the scene, the error varies. It is observable that a more detailed scene can be corrected better than a scene with large homogeneous areas.

### Influence of the Scene Properties

In the Figs. 3, 4 and 5, we show the influence of different kinds of cluttered regions on our up-scaling scheme. We used the approach of Freedman and Fattal [FF11] as the predictor and compare the predicted image with

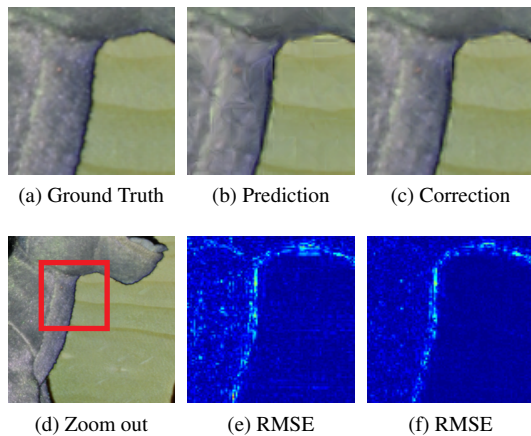


Figure 3: Direct component of a stuffed toy in front of a homogeneous background. The high frequencies of the fur are corrected more than the low frequencies of the background.

the results of the proposed correction scheme. Fig. 3 shows a close-up of a stuffed toy in front of a homogeneous background. As can be seen, our scheme reproduces the fine fur structure more detailed while in the predicted image this region looks smeared out. The homogeneous background is reproduced well with both methods. In Fig. 4, two pieces of chalk wrapped in paper in front of a printed box are shown. Here, in contrast to the stuffed toy, the fine details are not due to fine 3-dimensional structures. The letters on the box and the printed patterns of the paper wrapping the chalk are reconstructed in more detail with our correction. Fig. 5 shows small wooden figures with small scale painting and thin ornamentalations. The edges can be reproduced more precisely with the correction scheme compared to the prediction.

These three examples show that the quality of the correction is independent of the scene objects. Only the frequencies in the resulting image of the scene objects are responsible for the impact of the improvement. Fine structures, i.e. high frequencies, in the image lead to a large improvement in quality while areas with low frequencies cannot be corrected much more.

### Improvement of the Correction

The up-scaling of the five exemplary scenes was done with three different methods to evaluate the influence of the prediction to the correction. Nearest neighbour interpolation, bicubic interpolation and local self-examples from [FF11] were used as predictors for the up-scaling scheme. Fig. 6 shows a comparison between the RMSE for the different prediction methods and the afterwards corrected images. It can be seen that the correction always improves the predicted image, but depending on the quality of the prediction the correction

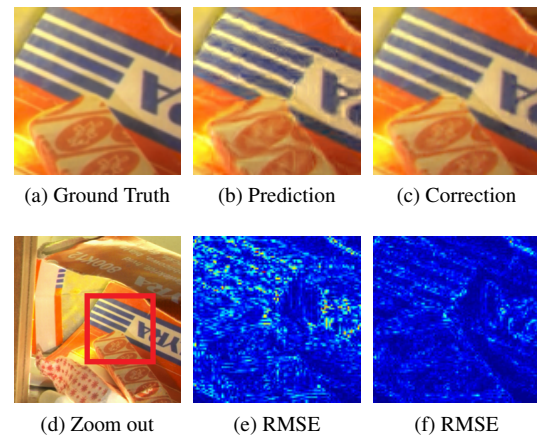


Figure 4: Global component of two pieces of chalk wrapped in paper in front of a printed box. Independent of the 3D object structure, the high frequencies are corrected much better.

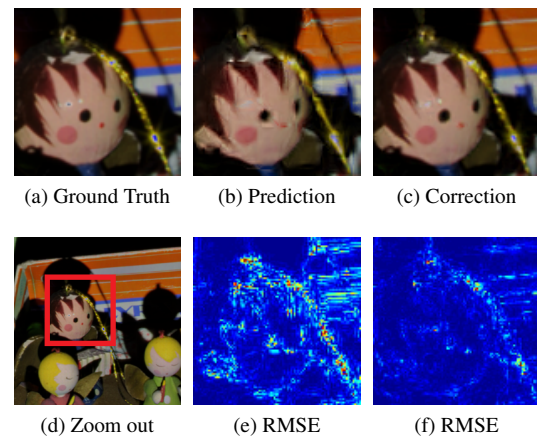


Figure 5: Direct component of small wooden figures in front of a printed box. The correction between different objects is corrected very good. High frequencies are corrected independently of the objects.

cannot compensate the larger error. For example, the correction of the nearest neighbour interpolation, which has the largest error in all scenes, improves the error but still has a larger absolute error than a better prediction without a correction for the global illumination in scene 3 and 5. An additional conclusion is that our proposed correction improves prospective predictions that will have less errors.

Please note that the local self-examples approach produces a higher prediction error than bicubic interpolation for scenes 2 to 5. In contrast to scene 1, these scenes contain cluttered regions. As stated by Freedman and Fattal, their approach is not able of handling these kind of image region and the algorithm spuriously reconstructs edges there [FF11]. This behaviour causes larger error than bicubic interpolation with its smoothing ability.



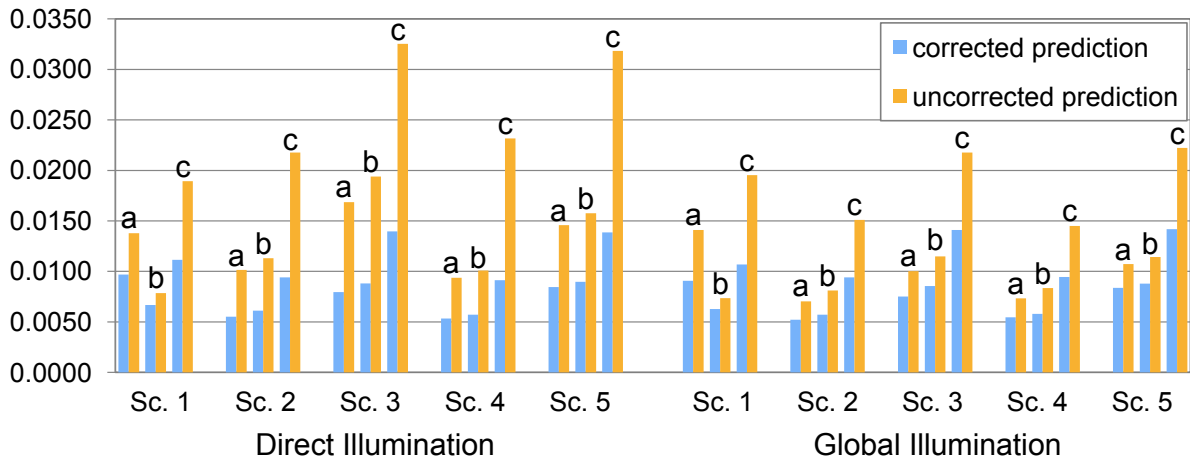


Figure 6: The sorted RMSE for (a) bicubic interpolation, (b) Local Self-Examples [FF11] and (c) nearest neighbour interpolation as predictors shows that the quality of the prediction influences the quality of the correction. The correction improves the quality of the prediction for direct and global illumination in all five scenes.

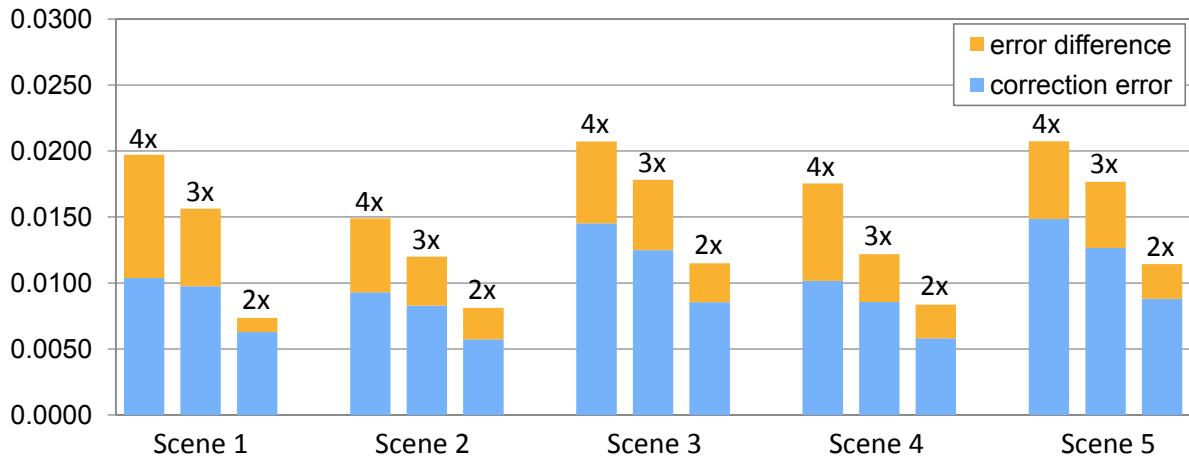


Figure 7: The dependency of the up-scaling factors is shown. For each scene the factors 4x, 3x, 2x are compared (left to right bar). A higher scaling factor leads to a higher error difference between prediction and correction. The global illumination images of the five test scenes are upscaled using [FF11].

### Influence of the Up-Scaling Factor

The up-scaling factor is another parameter of influence. Fig. 7 compares the RMSE for the scaling factors of 4x, 3x and 2x from left to right for each global illumination image of the five scenes. The single-image up-scaling method of [FF11] is used as predictor for this comparison. The absolute error introduced by the prediction is higher for larger scaling. But, it can be seen that the improvement of the reconstruction error by our correction scheme gets better for higher scaling factors in all scenes. As our algorithm works in an iterative manner, for higher up-scaling factors, more refinement steps are done. As in each step the predicted image is corrected we are able to better guide the up-scaling process.

### 5.2 Evaluation of Patterns

We used different structured-light patterns for the separation of the illumination component in order to evalu-

ate their applicability regarding our up-scaling scheme. As the focus is on low acquisition time, only patterns were chosen where up to three acquisition (including the fully illuminated image) are needed. Thus, we tested vertical stripes, horizontal stripes, a combination of both stripe patterns and a high-frequency checker-board pattern where each square had the same dimensions as the width of the stripes. The single-image separation approach proposed by Nayar et al. [NKGR06] was used in order to determine the low-resolution direct and global image components (cf. left part of Fig. 2). As for the stripes patterns a blurring of edges within the direction of the stripes occur, we combined the separation results of both patterns in order to compensate for this effect. However, for this procedure an additional acquisition is required. Therefore, we introduced the high-frequency checker-board pattern. Here, the same separation methods were applied as for horizontal

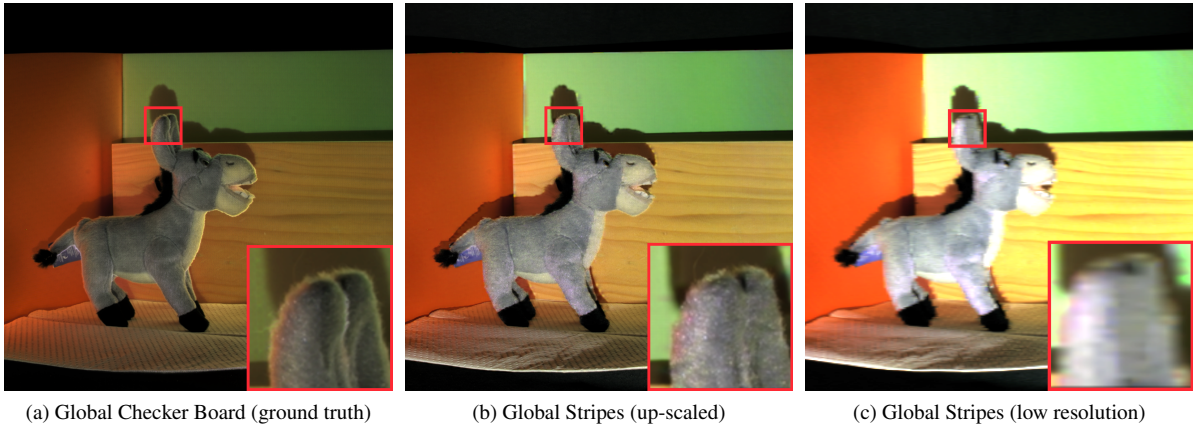


Figure 8: Comparison of the separation methods using checker-board (a) and stripes structured-light patterns (b). As can be seen, the stripes separation (c) suffers severe artefacts. Our up-scaling scheme is not capable of properly reconstructing the high-resolution images.

	vert. stripes	hor. stripes	comb. stripes	vert. checker	hor. checker	comb. checker	down-sampled ground truth
RMSE (LSE)	0.2219	0.2024	0.2085	0.3083	0.3099	0.3089	0.0140
$\ell_1$ -Norm (LSE)	29703	26350	27768	41835	42058	41945	1236
$\ell_2$ -Norm (LSE)	221.88	202.41	208.52	308.26	309.85	308.91	14.01
RMSE (BC)	0.2147	0.1826	0.1948	0.3020	0.3036	0.3015	0.0233
$\ell_1$ -Norm (BC)	28792	24059	26160	41043	41369	41066	1810
$\ell_2$ -Norm (BC)	214.72	182.55	194.84	302.03	303.63	301.49	23.30

Table 2: Different kind of patterns were used for a fast separation. The resulting images were fed into our up-scaling scheme with local self-examples (LSE) and bicubic interpolation (BC) as predictor.

and vertical stripes. In order to improve blurring artefacts, the separation results were combined, also. Tab. 2 shows the reconstruction error of our up-scaling scheme for each of these separation methods compared to the down-sample ground truth. As predictors we used bicubic interpolation and local self-examples [FF11]. Our experiments show that the reconstruction results differ depending on whether the down-sampled ground truth or the low-resolution images resulting from the stripes or checker-board patterns are used. As can be seen in Fig. 8, the stripes-pattern separation suffers severe reconstruction artefacts. Besides the wrong color reproduction in homogeneous regions, at diagonal edges stair-case artefacts are present. Furthermore, cluttered regions are subjected to noise. Although our up-scaling scheme is capable of reducing the artefacts in brighter image regions, it cannot recover the fine details. In general, the up-scaling error for the stripes patterns is smaller as for the checker-board. The choice of the predictor does not influence the up-scaling result significantly. Also, the combination of horizontal and vertical stripes could not improve the quality of the input images. By monitoring the weights for the prediction error distribution  $\alpha_d$ ,  $\alpha_g$ , we realized that they were close to 0.5 for all pixels of the direct and global images. This is no surprise, as the noise in both images cannot be reduced significantly by distributing the prediction error.

## 6 CONCLUSIONS

We presented a new constrained up-scaling approach for direct and global component images. Predictions based on single-image up-scaling techniques are corrected using a high resolution fully illuminated image. It can be seen that our method can visually improve cluttered regions of an image such as fur or fine structured ornamentations. The experiments showed that our algorithm scales with the quality of the prediction as even well predicted images can be improved further. The iterative manner of the method is also capable of improving single-image up-scaling approaches for large up-scaling factors.

Our method is applicable when the acquisition full resolution images using many structured-light patterns is not possible due to limited time-constraints. This is the case for capturing dynamic scenes in Computer Vision tasks, such as biometry of human faces or mobile autonomous navigation. Here, only two images are required. However, by acquiring more than one image our method is prone to motion artefacts and misalignment of the two images. Thus, it might be necessary to apply optical flow or motion compensation. Also, our up-scaling scheme poses additional computational complexity which might not be suitable for real-time analysis of the acquired data.



Furthermore, when using structured-light patterns or variations known in literature for fast acquisition, the quality of the resulting low-resolution images is not sufficient. Obviously, these separation methods are strongly influenced by wrong color reproduction and stair-case artefacts. Thus, these kind of patterns produce fundamentally different direct and global component images than methods which work directly at the resolution of the acquired images.

In future work, we study the impact of different light patterns on the separation results. This will give us useful insight about how to solve the above mentioned discrepancies between low-resolution images and ground-truth data.

## 7 ACKNOWLEDGMENTS

This work was funded by the German Research Foundation (DFG) as part of the research training group GRK 1564 'Imaging New Modalities'.

## 8 REFERENCES

- [AB12] L. An and B. Bhanu. Image super-resolution by extreme learning machine. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pages 2209–2212, 2012.
- [ABA01] C. Atkins, C. Bouman, and J. Allebach. Optimal image scaling using pixel classification. In *Proc. Int. Conf. Image Processing*, volume 3, pages 864–867, 2001.
- [BGS02] S. Battiato, G. Gallo, and F. Stanco. A locally adaptive zooming algorithm for digital images. *Image and Vision Computing*, 20(11):805–812, 2002.
- [CHL05] M.-J. Chen, C.-H. Huang, and W.-L. Lee. A fast edge-oriented algorithm for image interpolation. *Image and Vision Computing*, 23(9):791–798, 2005.
- [EV07] M. Ebrahimi and E. Vrscay. Solving the inverse problem of image zooming using self-examples. In M. Kamel and A. Campilho, editors, *Image Analysis and Recognition*, volume 4633 of *Lecture Notes in Computer Science*, pages 117–130. Springer Berlin Heidelberg, 2007.
- [Fat07] R. Fattal. Image upsampling via imposed edge statistics. *ACM Trans. Graph.*, 26(3), 2007.
- [FF11] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Trans. Graph.*, 30(2):12:1–12:11, 2011.
- [FJP02] W. Freeman, T. Jones, and E. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.
- [GA08] A. Giachetti and N. Asuni. Fast artifact free image interpolation. In *Proc. BMVC 2008*, 2008.
- [GA11] A. Giachetti and N. Asuni. Real-time artifact-free image upscaling. *IEEE Trans. Image Processing*, 20(10):2760–2768, 2011.
- [GADTH97] M. Gharavi-Alkhansari, R. DeNardo, Y. Tenda, and T. S. Huang. Resolution enhancement of images using fractal coding. *Proc. SPIE 3024, Visual Communications and Image Processing*, 3024:1089–1100, 1997.
- [GBI09] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Proc. IEEE Int. Conf. Computer Vision*, pages 349–356, 2009.
- [GKGN11] J. Gu, T. Kobayashi, M. Gupta, and S. K. Nayar. Multiplexed illumination for scene recovery in the presence of global illumination. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 1–8, 2011.
- [GL12] J. Gu and C. Liu. Discriminative illumination: Per-pixel classification of raw materials based on optimal projections of spectral brdf. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 797–804, 2012.
- [KK08] K. Kim and Y. Kwon. Example-based learning for single-image super-resolution. In G. Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 456–465. Springer Berlin Heidelberg, 2008.
- [LHL11] B. Langmann, K. Hartmann, and O. Lof-feld. Comparison of depth super-resolution methods for 2d/3d images. *Int. Journal of Computer Information Systems and Industrial Management Applications*, 3:635–645, 2011.
- [LHL12] B. Langmann, K. Hartmann, and O. Lof-feld. A modular framework for 2d/3d and multi-modal segmentation with joint super-resolution. In A. Fusiello, V. Murino, and R. Cucchiara, editors, *Computer Vision - ECCV 2012. Workshops and Demonstrations*, volume 7584 of *Lecture Notes in Computer Science*, pages 12–21. Springer Berlin Heidelberg, 2012.
- [LLK08] M. Lindner, M. Lambers, and A. Kolb.

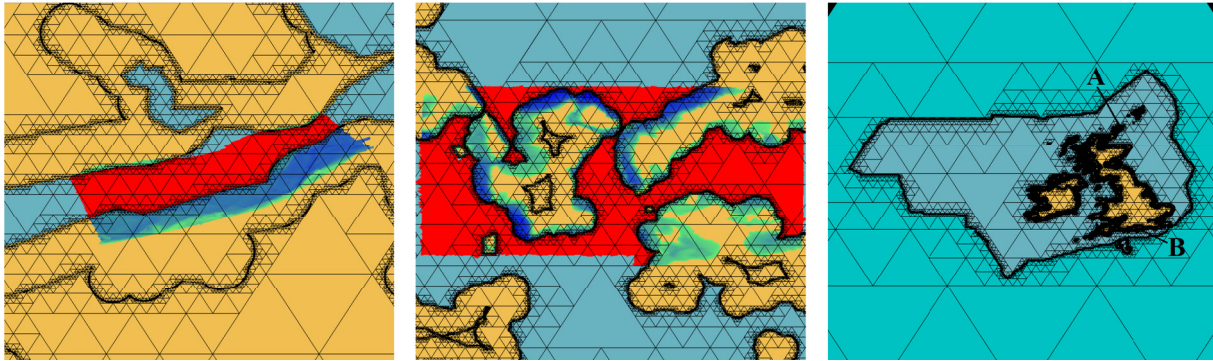
- Data fusion and edge-enhanced distance refinement for 2d rgb and 3d range images. *Int. J. on Intell. Systems and Techn. and App. (IJISTA), Issue on Dynamic 3D Imaging*, 5(1):344 – 354, 2008.
- [LO01] X. Li and M. Orchard. New edge-directed interpolation. *IEEE Trans. Image Processing*, 10(10):1521 –1527, 2001.
- [MYR10] Y. Mukaigawa, Y. Yagi, and R. Raskar. Analysis of light transport in scattering media. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 153–160, 2010.
- [NG12] S. K. Nayar and M. Gupta. Diffuse structured light. In *Proc. IEEE Int. Conf. Computational Photography*, pages 1–8, 2012.
- [NKGR06] S. K. Nayar, G. Krishnan, M. D. Grossberg, and R. Raskar. Fast separation of direct and global components of a scene using high frequency illumination. *ACM Trans. Graph.*, 25(3):935–944, 2006.
- [ORK12] M. O’Toole, R. Raskar, and K. N. Kutulakos. Primal-dual coding to probe light transport. *ACM Trans. Graph.*, 31:39:1–39:11, 2012.
- [RRC12] D. Reddy, R. Ramamoorthi, and B. Curless. Frequency-space decomposition and acquisition of light transport under spatially varying illumination. In *Computer Vision - ECCV 2012*. Springer Berlin Heidelberg, 2012.
- [SSU08] N. Suetake, M. Sakano, and E. Uchino. Image super-resolution based on local self-similarity. *Optical Review*, 15:26–30, 2008.
- [STDT08] S. Schuon, C. Theobalt, J. Davis, and S. Thrun. High-quality scanning using time-of-flight depth superresolution. In *Proc. IEEE Conf. on Comp. Vis. & Patt. Recogn.; Workshops (CVPRW)*, pages 1 –7, 2008.
- [SW04] D. Su and P. Willis. Image interpolation by pixel-level data-dependent triangulation. *Computer Graphics Forum*, 23(2):189–201, 2004.
- [SXS08] J. Sun, Z. Xu, and H.-Y. Shum. Image super-resolution using gradient profile prior. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1 –8, 2008.
- [TM96] S. Thurnhofer and S. K. Mitra. Edge-enhanced image zooming. *Optical Engineering*, 35(7):1862–1870, 1996.
- [TTT06] Y.-W. Tai, W.-S. Tong, and C.-K. Tang. Perceptually-inspired and edge-directed color image super-resolution. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1948 – 1955, 2006.
- [WXM<sup>+</sup>13] L. Wang, S. Xiang, G. Meng, H.-y. Wu, and C. Pan. Edge-directed single image super-resolution via adaptive gradient magnitude self-interpolation. *IEEE Trans. Circuits and Systems for Video Technology*, PP(99):1, 2013.
- [YWL<sup>+</sup>12] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE Trans. Image Processing*, 21(8):3467 –3478, 2012.
- [YXXY12] L. Yu, H. Xu, Y. Xu, and X. Yang. Robust single image super-resolution based on gradient enhancement. In *Proc. Signal Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1 –6, 2012.
- [YXYC12] Q. Yan, Y. Xu, X. Yang, and K. Chen. Image super-resolution based on a novel edge sharpness prior. In *Proc. Int. Conf. Pattern Recognition (ICPR)*, pages 1056 –1059, 2012.
- [ZDLL12] L. Zi, J. Du, M. Liang, and J. Lee. A perception-motivated image interpolation algorithm. In *Proc. World Congress on Intelligent Control and Automation (WCICA)*, pages 4754 –4759, 2012.

# Trixel Buffer Logic for I/O Bound Point in N-Polygon Inclusion Tests of Massive Bathymetric Data

João Fradinho Oliveira<sup>1</sup> Marek Ziebart<sup>2</sup> Jonathan Iliffe<sup>3</sup> James Turner<sup>4</sup> Stuart Robson<sup>5</sup>

<sup>1</sup>C3i/Instituto Politécnico de Portalegre, Portalegre, Portugal. jfoliveira@estgp.pt

<sup>2,3,4,5</sup> Department of Civil Environmental and Geomatic Engineering, University College London, U.K.  
{m.ziebart, j.iliffe, james.turner, s.robson}@ucl.ac.uk



**Figure 1.** I/O bound point in N-polygons inclusion queries - “Valid” points in hues of blue/green (colour coding the sea floor depth) are inside the project polygon limit area AND outside the offset coastal inner polygon AND inside the offset costal outer polygon. “Invalid” points in red are inside the project polygon, outside the offset coastal inner polygon BUT outside the offset coastal outer polygon. Left: result of 1.757 billion bathymetric point queries (Solent[B]1h36m DuoCore2.5Ghz, details in Section4) Center: 61 million queries (Kirkwall[A]~ 3.36min). Right: project polygon limits in dark blue and quadtree root in light blue.

## ABSTRACT

“Trixel Buffers is a new spatial data-structure for fast point in multiple polygon inclusion queries. The algorithm utilizes a pre-processing step in which the inside/outside status of a quadtree’s leaf triangles without polygon geometry is pre-computed automatically; at run-time point queries lying within these triangles simply inherit their inclusion status. If a point query lies in a leaf triangle enclosing polygon vertices or crossing edges, a ray is fired from the point towards the triangle center whose polygon inclusion properties has also been pre-computed: rules are then applied to the intersection count and center-point properties to infer the polygon inclusion status. Our main contribution is that rays need not be followed until the polygon limits, and consequently the algorithm is I/O bound with shallow trees. It took 1h36m rather than days of using a standard ray test to determine the multiple polygon (~270,000 line segments) inclusion of 1.75 billion points on a 2.5GHz DuoCore computer.

## Keywords

Point in polygon test, point-location problem, trixel buffer logic, point buffers, bathymetric data.

## 1. INTRODUCTION

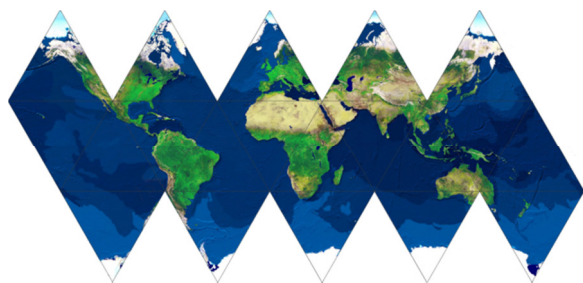
The seemingly simple task of determining whether a point is inside or outside a given polygon, or indeed a set of several polygons at the same time is an integral task within many diverse applications, such as a geometric editing/polygon selection, climate simulation, and interactive computer graphics. A

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

popular solution used for example in ArcGIS is simply to render polygons and lookup the rendered pixel attributes (polygon IDs) at our mouse click position. This approach works well by relying on the user and user interface to zoom interactively to a high level of detail in a local region of interest if they wish more accuracy in their polygon selection. Unfortunately, this strategy would not work well in the context of determining the point-in polygon status of large volumes of data/query points from a file such as bathymetric survey data from the sea floor as the user cannot afford to set the zoom level/center the view location for each point to obtain accurate enough results from rendering. Furthermore, high detail polygons can rapidly make such a rendering

approach for point polygon inclusion cumbersome; in addition files containing the point queries can be streamed from different users over a network and be of arbitrarily locations, making the position and zoom levels of such renderings difficult to optimize.

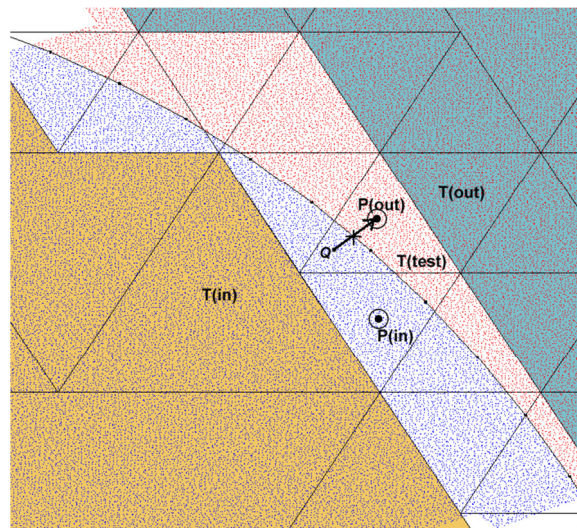
Several algorithms model polygon geometry directly offering resolution independence and higher accuracy results. A ray test [Tay94a, Prep85a] handles convex and concave polygon but without a hierarchical spatial tree will be comparatively slow as is the winding number method that still needs to test/reject several polygon segments. Quadtrees [Sam90a] can be used to solve the point-location problem rapidly [Sar86a], [Edel86a, Kir83a, Pat06a], and depending on the application scenario one could use rectangular quadtrees [Pov04a], or triangular quadtrees [Fek90a] if mapping the whole globe. To avoid distortions at the poles, each of the 20 equilateral triangles of an unrolled icosahedron is a root quadtree (Figure 2).



**Figure 2. Icosahedron, 20 equilateral root triangles. Courtesy [Ham12a].**

The idea of solving point-in polygon queries by using quadtree leaf cells to buffer and cache pre-computed polygon-inclusion results is not new [Pov04a], Trixels [Fek90a]. However several problems arise when addressing point-in polygon queries of massive bathymetric data using complex high detail polygons. The first problem is memory consumption, which scales badly with increasing the vertex spacing resolution of the polygons. I/O bound performance of reported strategies for finding the polygon inclusion status of point queries near polygon geometry require that the tree was subdivided until the size of leaf cells matches the resolution of the polygons. If considering applications running on mobile devices with even more limited memory this problem is even greater. Secondly, existing methods require the manual setting of a known interior polygon position to propagate results during the pre-processing stage. When dealing with thousands of offset polygons in the context of our application this becomes cumbersome or impractical to ask the user, as the size of many polygons derived automatically from real data can be very small (Fig 1, center). Thirdly our application must be robust to deal with convex and concave polygons with arbitrary vertex ordering resulting from merging/importing of several polygon shape files from different tools.

In this article we present a hierarchical spatial database (Trixel Buffers) solution to these problems, where leaf triangles void of geometry are termed trixel buffers; point queries lying in a trixel buffer simply inherit the polygon-inclusion status of that triangle (for example triangle  $T(in)$  and  $T(out)$  in Fig.3. In contrast a point query ( $Q$ ) lying in a leaf triangle that has polygon vertices or crossing edges  $T(test)$  requires an additional ray test with the known polygon inclusion point buffer  $P(out)$ . Trixel Buffers were designed as part of the Vertical Offshore Reference Frame (VORF) project [Ili06a], which modeled the datum surfaces used for spatial data on land and at sea around the UK and Ireland. The work carried out enables transformations between datums used by satellite positioning systems, marine datums used for bathymetric data, and land datums used for topographic data. With 17 different land datums and multiple complex polygons defining navigable rivers and harbors, robust and efficient position tests need to be performed on each point in very large datasets.



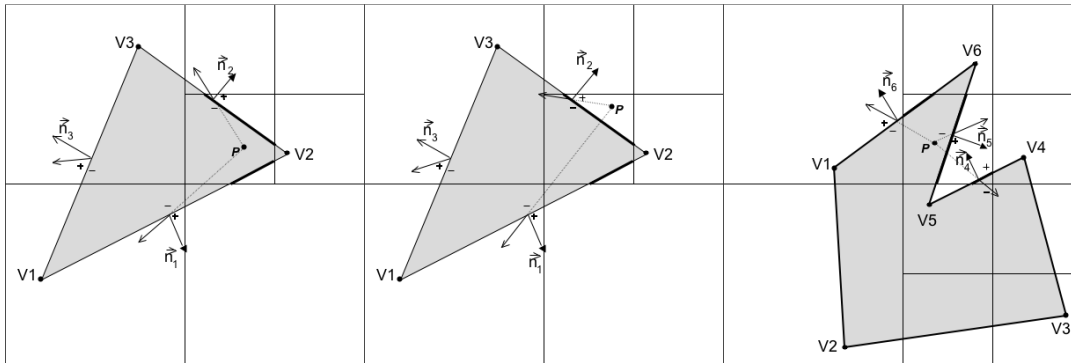
**Figure 3. Trixel buffer logic.**

The main concept of our algorithm is in the case of a query point that lies in a leaf triangle with polygon geometry, to cast a ray from the query point to its triangle center and apply trixel buffer logic to the center point's pre-computed polygon inclusion status and the number of intersections of the ray with any polygon geometry in the leaf triangle to infer the inside/outside status of the query point, rather than continue to trace a ray until it exits the polygon or quadtree.

#### Contributions:

We present a new point in N-polygon algorithm that:

- Extends the Gauss-Jordan theorem to work efficiently with hierarchical spatial trees.
- Is I/O bound with massive data sets, requiring relatively shallow trees.
- Extracts interior/exterior of polygons automatically.



**Figure 4. Inner product with convex polygons (left & center) and with concave polygons (right). left: BOTH inner products with  $n_1$  and  $n_2$  are positive,  $P$  is inside; center: ONE OF the inner products with  $n_2$  is negative,  $P$  is outside. right: ONE OF the inner products ( $n_4$ ) is negative and  $P$  is incorrectly labeled as out.**

- Handles convex and concave polygons with arbitrary ordering.
- Is extendable to other dimensions.
- Can be used to extend existing quadtree methods by calculating the polygon inclusion status of point buffers (which are infinitely small trixel buffers) and applying our presented rules.

We briefly review related work in Section 2. We note that we use the words ‘trixel buffer’ and ‘triangle nodes’ interchangeably throughout the paper, and the words ‘coastal line’ in reference to ‘polygon lines that represent real coast lines’. In section 3, we show how the quadtree is used to create trixel buffers automatically, and how our new short ray-strategy is used in conjunction with trixel buffers to determine which polygons a point lies within. In section 4 we present results; specifically we use a brute-force ray algorithm [Tay94a] to inspect and validate our polygon inclusion results; we also show that our method does not require the resolution of leaf triangles to match the resolution of the input polygons to achieve I/O bound performance. In section 5 we present a discussion, and conclude in section 6.

## 2. PREVIOUS WORK

A vast body of literature exists in computational geometry for the detection of whether a point is inside a convex hull, or inside a generic planar polygon [Berg97a, Prep85a, Hai94a, Edgel86a]. Perhaps the simplest way to determine whether a point is inside or outside of a polygon is to fire a ray horizontally from the point in question to  $+$  or  $-$  infinity and apply the Jordan curve theorem on the number of intersections of the ray with the edges of the polygon. If the number of intersection is odd, the point is deemed inside, if the number is even the point is deemed outside. Special care [Prep85a] is taken for rays that pass through the vertices of a polygon, horizontal edges are ignored, and if the ray intersects a vertex, and the vertex has the largest ordinate of the edge the intersection is counted,

otherwise it is ignored. Even though many polygon edges can be trivially rejected from any intersection by checking if both ordinates of the edge vertices are both greater or both smaller than the ray’s ordinate, this algorithm has a query complexity of  $O(N)$  as it checks every edge of a polygon before determining to test it for intersection or not. Fast solutions exist for the planar point location problem: given a planar subdivision of space, the task to establish which cell or polygon contains our query can be achieved in  $O(\log(N))$  using persistence search trees [Sar86a], fractional cascading [Edel86a], and triangulation refinement [Kir83a]. Recently sub-logarithmic complexity for queries has been achieved with support for dynamic planar subdivisions [Pat06a]. In particular Kirkpatrick [Kir83a] shows  $O(N \log(N))$  preprocessing time with  $O(n)$  storage using hierarchical triangle subdivisions. Similarly we use a triangular quadtree in this paper for the point location problem, and modify it to support a ray strategy for solving the point-in polygon problem. Poveda et al. [Pov04a] use a square quadtree to buffer the polygon inclusion status in cell nodes void of geometry, and report I/O bound results with quadtree leaf buffers whose length matches the vertex spacing resolution of a convex polygon set. For query points in a cell with geometry, they use an inner product test (Fig.4-left&centre), but to our understanding unfortunately this test will not work in the case of concave polygons (Fig.4-right).

This method also requires the manual seeding of a known interior point. As mentioned earlier Fekete [Fek90a] uses 20 equilateral triangles of an icosahedron as root nodes of triangular quadtrees instead of rectangular quadtrees to avoid distortions at poles [Ran02a, Oli06a]. Fekete stressed the need to combine a spherical visualization representation with the actual data coordinates for global simulation of the atmosphere [Ran02a]. Hence the length of the position vectors, defined by the triangle edge midpoints are adjusted during subdivision to a set radius or property, thus creating a spherical quadtree.



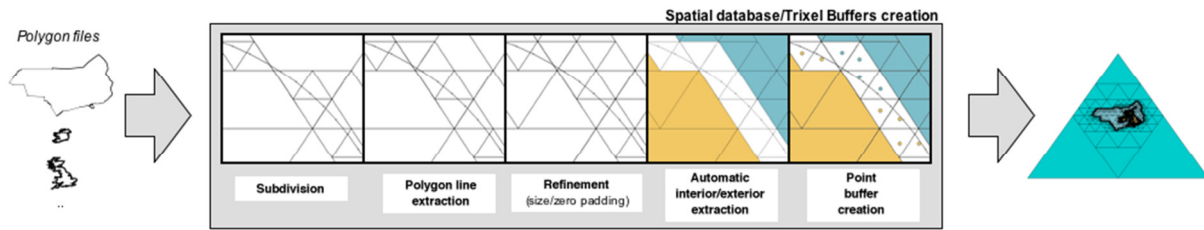


Figure 5. Overview of the construction of the Trixel Buffer spatial database.

Fekete does not address directly point-in polygon tests in his data structure, but uses a technique called connected component labeling that uses connectivity information stored in a tree node to access and propagate the inside or outside results of cells to adjacent nodes. A seed cell is manually chosen inside a landmass, and the result is propagated to the limits of the landmass.

	[Tay94a]	[Fek90a]	[Pov04a]	Trixel Buffers
Manual (m) / automatic (a) interior extraction	a	m	m	a
Convex Polygons	y	n/a	y	y
Concave Polygons	y	n/a	n	y
Tree depth for I/O Performance w/ polygon data of Fig.1	n/a	n/a	16	12

Table 1. Comparison of properties of existing methods for point in polygon tests of massive bathymetric data.

One can envisage that points inside cells void of geometry simply inherit the extracted landmass IDs. Point queries inside cells with geometry could use the point buffers presented in this paper to determine landmass inclusion. Table 1 compares the properties of existing methods for massive bathymetric tests.

### 3. TRIXEL BUFFER CREATION

Our system is built with five main steps (Fig.5, middle). The first step creates a hierarchical spatial database through quadtree triangular subdivision. This step solves the point location problem. The second step termed polygon line extraction adds further triangles in areas of passing edges to ensure that all polygon segments can be readily referenced. Further refinement of the tree in step 3 maximizes the area void of geometry. In step 4 the polygon inclusion status of triangles void of geometry is computed automatically through the interior/exterior extraction algorithm. Finally in step 5 the center

point inside each leaf triangle with geometry has its interior/exterior status calculated.

#### Subdivision

Although we use a triangular quadtree rather than squares, the process of subdivision is similar. In the context of our project, one of the twenty base equilateral triangles of the icosahedron sufficed to enclose our survey data. The construction of the quadtree starts with a simple  $O(N)$  pass on every polygon vertex in order to establish the maximum and minimum coordinates of the set. An equilateral triangle which encloses all the data can then be computed. A subsequent subdivision process, using the three middle points of each of the triangle's edges, creates four smaller equilateral triangles. In our implementation, for precision purposes the width and the height of the root triangle are stored once separately. The width and height of a quadnode of any level can be calculated by a single division made to the original width/height of the root triangle by powers of two representing the subdivision level.

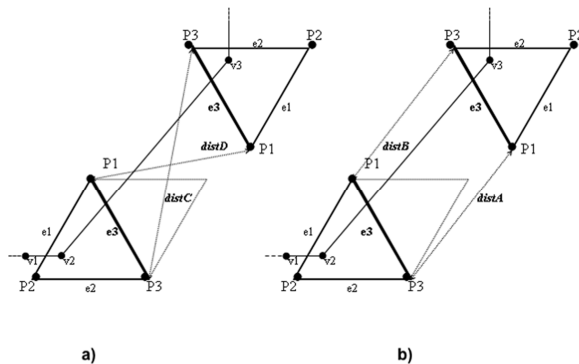
This process of subdivision is repeated until the set maximum depth level is reached. In order to maximize the number of query points not requiring geometric tests at run-time, we first compute the average vertex spacing distance of the polygon set, the limit of the subdivision depth can then be set by the length of the triangle edge being subdivided. If the length is smaller than the average spacing we stop the subdivision. Standard recursive spatial data structures such as the quadtree set recursion limits such as the maximum number of input points allowed in the deepest triangles and/or the maximum tree depth allowed. Unfortunately this strategy can yield leaf node triangles that are very large. We record the width and height of the smallest quadnode found, and in a second pass, we further refine leaf nodes that have geometry if their size is larger than the smallest triangle found.

#### Polygon line extraction and refinement

Whilst the subdivision step is centered on the polygon vertices, a water tight front of leaf nodes covering the complete geometry, including crossing edges is required before rays from any position in the tree can reliably test the polygons for inclusion, independently of the spacing of the vertices of the polygon. To detect crossing edges, and minimize the size of these areas, we insert leaf nodes in the tree in



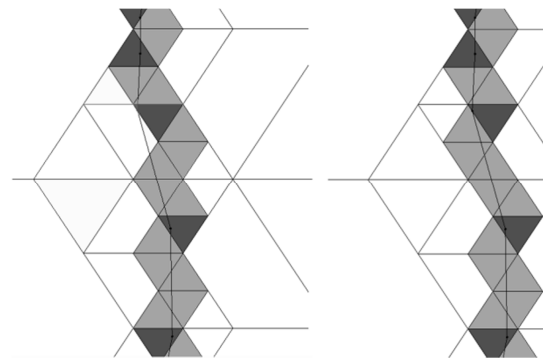
areas covering long edges; in addition these nodes/triangles also store the ID of one of the vertices of the long edge, this ID can then be used later to build the edges for ray intersection tests. To build this connected front a procedure with a result similar to that of Bresenham's line drawing algorithm is performed, starting by looking up the leaf nodes containing each endpoint of each polygon edge. We refer to the leaf node containing the first vertex of an edge as the starting leaf node, and the leaf node containing the endpoint vertex as the target leaf node. If the two vertices are contained in the same leaf node, no leaf nodes need be inserted, otherwise we systematically insert leaf nodes adjacent to the node containing the starting vertex of the edge, until the inserted leaf node is adjacent to the leaf node containing the end vertex of the edge. Here adjacency requires that the leaf nodes share two vertices. Since we do not store adjacency/connectivity information, we retrieve adjacent triangles covering the polygon edge, using a three step process.



**Figure 6. Minimum sum distance between leaf triangle edges. The starting leaf triangle containing the first vertex V2 of the edge V2-V3 has minimum sum distance ( $\text{distC} + \text{distD(a)} > \text{distA} + \text{distB(b)}$ ) between the triangle edge e3 of the start triangle and the edge e3 of the target triangle.**

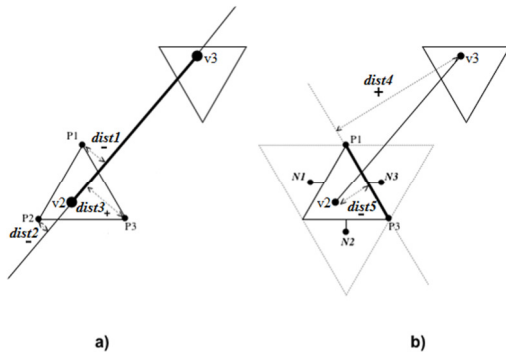
The first part establishes which edge from the starting leaf node has the smallest summed vertex distance to any edge of the target leaf node (Fig.6-a&b)). However, if one were to insert an empty node/triangle adjacent to the edge simply determined to be of shortest distance to the target triangle, no guarantee could be given that the polygon edge was completely contained by the inserted leaf nodes (Fig.7-left). This distance calculation is however useful to determine the stopping condition of zero distance between adjacent and target triangles. The second part of the algorithm finds which edge(s) of the starting triangle are candidates for inserting the adjacent empty leaf node. The third part determines which of the potential two candidates is chosen to insert the adjacent node. A polygon edge can at most intersect a triangle in two of its three edges. For

example Fig.6-a) shows that the edge v2-v3 intersects the starting triangle in one edge (e3), and intersects an adjacent triangle in two edges.



**Figure 7. Polygon line extraction. - Dark grey triangles contain vertices of the polygon. Light grey triangles on the left image denote incorrectly inserted empty triangles along the edge using the minimum sum distance to the target triangle, light grey triangles on the right image were inserted using the polygon edge geometry (Fig.8) instead.**

We note that once the leaf nodes of the polygon edge vertices are found (recursive look-up of the tree), the line connecting the polygon edge vertices is guaranteed to intersect both the starting and target leaf node triangles if the vertices are in different leaf triangles (if adjacent no triangles are inserted, if separated by an area void of leaf nodes, new empty leaf triangles are inserted in the path of the edge, if a leaf node is already present in the path, the ID of the passing edge is stored in the node instead) or can have its two vertices in the same leaf triangle (start and target triangle are the same, no adjacent triangle needs to be inserted for this vertex pair as any passing ray will retrieve both edges connected to each vertex). In order to narrow down the candidate edges to two, we determine which triangle edges the polygon edge might intersect. Specifically we compute the signed distance of each vertex of the triangle to the polygon edge, the triangle edges whose vertex distances have different signs indicates that the triangle edge is intersected by the line going through the polygon edge. Fig.8-a) shows that the triangle edge P1-P3 and the triangle edge P2-P3 have different point distance signs, although only the edge P1-P3 is intersected by the polygon edge V2-V3. The final step of the coastal line extraction algorithm is to determine which of the two candidate edges, is the edge in which the system is going to insert an adjacent empty node. For this effect we use an efficient ray rejection technique [Xu03a], where the endpoints of this polygon edge are tested against the candidate edges. The triangle edge that yields different signs in the edge endpoint distance test is selected. For example, the triangle edge P1-P3 of Fig.8-b) is the only edge with different distance signs.



**Figure 8. Polygon line extraction using the polygon edge geometry.**

Finally we add the ID of the first endpoint of the polygon edge to the adjacent empty node (this enables for point queries in the adjacent triangle to retrieve and build the edge that crosses it), if there was an adjacent node already with geometry the vertex ID is added to it nevertheless. The adjacent triangle then becomes the starting triangle. The polygon edge is used as before with the new triangle, however the starting vertex (v2) used for the step (Fig.6-a) and b)) becomes the offset point (N3 of Fig.8-b)). This process is repeated until the starting triangle is the same as the target triangle or is adjacent to the target triangle (Fig.7-right)). To make sure any point query location has a non-overlapping leaf node, a refinement procedure is done, specifically, every node from the root is checked to see if a subnode exists with adjacent null node pointers, new empty triangle nodes representing smaller areas void of geometry then replace the null pointers.

### Automatic interior/exterior extraction

Most point in polygon methods require manual seeding of a node known to be interior to the polygon. Our algorithm calculates automatically the polygon inclusion status of nodes void of geometry using the following rule:

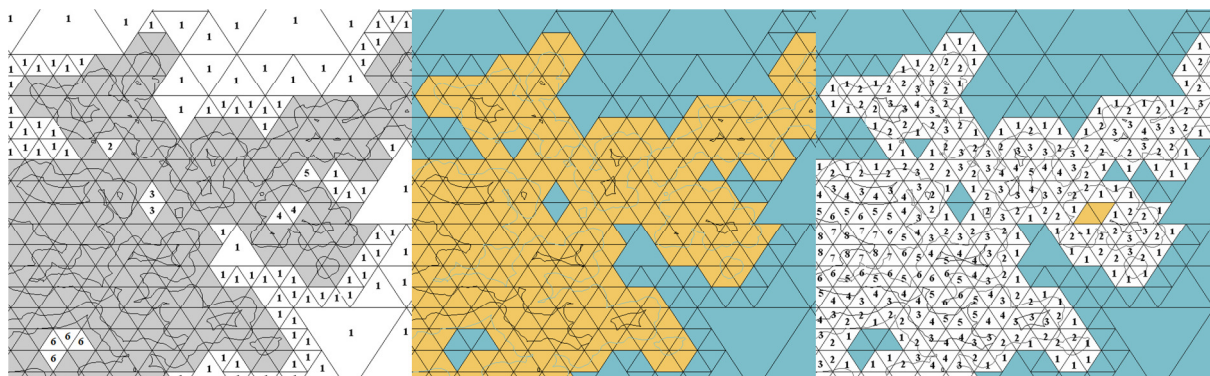
**Rule 1** – Any point within a node void of geometry has the same polygon inclusion status as any other point within the node. If one fired a ray from any point of the node towards infinity in any direction, and the node was indeed completely interior to a polygon, then the Gauss Jordan rule will give the same result for that polygon, independent of the number of intersections. Special care is taken for rays that pass through the vertices of a polygon, horizontal edges are ignored, and if the ray intersects a vertex, and the vertex has the largest ordinate of the edge the intersection is counted, otherwise it is ignored.

Connected adjacent leaf nodes void of geometry are grouped and the inside/outside status of one of the cells in the group is calculated with rule 1, the result

is then copied to all the other elements in the group using following rule:

**Rule 2** – Rule 1 can be generalized to the outline of triangle strips void of geometry or more arbitrary connected contour shapes, as long as the single ray being fired does indeed traverse to infinity or beyond the known pre-calculated bounds of the polygons being tested.

We use a flood-fill algorithm to group connected adjacent triangles (Fig.9-left)) as follows: we build an array of pointers to all the leaf nodes void of geometry in the tree, and we zero a group counter and the group attribute of each leaf. We sort the array in increasing triangle size (a small triangle will always have only one adjacent neighbor on one side, rather than two or more if starting with a larger triangle first), and visit each element of the array that has no group assigned, we increment the group counter each time we find a triangle that has not had a group ID assigned and assign the current group number to the triangle, we then retrieve the three adjacent triangle neighbors (left, right and vertical) of the triangle. If one of the neighbors has no geometry and has not been assigned a group, we assign the current group counter value to it and push it on to a stack. While the stack is not zero we keep removing the last element of the stack before proceeding with the next element of the sorted array. Once all leaf triangles of the array have been visited, we sort the array again this time on group ID, the first leaf node of a group found in the array is then ray tested as described below and the result is copied to all the array element of the same group. As mentioned earlier, each connected group has one leaf node void of geometry ray tested, a horizontal ray from the middle of a triangle is fired towards a point outside the tree. If the ray passes a triangle that has polygon vertices or flagged geometry, tests for intersection are carried out on polygon edges. Specifically each vertex contained or flagged in a triangle is looked up, and two edges are built by retrieving the two other connected vertices. These connected vertices can lie arbitrarily far from the ray path, and the ray might only intersect the edges within adjacent triangles. In order to not test an edge more than once during the course of the ray path and to keep track of whether an edge has already been counted as a hit, each vertex of our polygon set has two flags reflecting the intersection status of the two edges that connect to it. These flags can have a mark of 0, 1 or 2, depending on whether the edge has not been tested (0), tested and hence not requiring further tests (1), or tested and with a hit that has been counted (2). If a ray crosses a triangle with no geometry, that triangle is skipped, and the next adjacent triangle in the ray path is retrieved, until the ray exited the tree. All the nodes that were visited have the flags of their vertices



**Figure 9. Left: Flood fill grouping of triangles void of geometry, numbers represent each connected group. Center: Interior/exterior extraction results. Right: Pre-computation for point buffer creation, numbers represent the smallest number of triangles that need to be traversed to find an adjacent triangle void of geometry, this will be used to establish the shortest route for a ray when determining the point buffer interior/exterior status.**

zeroed, before proceeding with any more triangle/polygon inclusion tests. During ray tests a separate intersection count is kept for each polygon ID the ray crosses. If the final intersection count for a particular polygon ID is odd, the starting selected triangle is deemed to be inside that polygon, and the ID of that polygon is registered in the triangle. If the count is even, the triangle is outside of that polygon, and the polygon ID for that count is not stored. Note that it is possible for a triangle to be inside more than one polygon, or to be outside a tested polygon but inside another polygon. Fig.1-c) and Fig.9-centre) shows extracted landmasses in brown that are all within a sea polygon in dark blue. It is interesting to note, that since we always retrieve the two edges that are connected to a vertex our ray intersection is not adversely affected in the presence of clockwise and counter clockwise polygons. Any duplicate polygon would have different polygon IDs, and hence does not inadvertently affect the inside/outside counting test. Table 3 shows spatial and memory statistics of Trixel Buffers. We note that the average vertex spacing of our 270,000 polygon edge data set is 136 meters and that this spacing is almost the same everywhere. A strategy whose run-time performance relies exclusively on minimizing the number of geometric primitives to test on cells with geometry would require a tree of depth 16 with leaf edges of 100 meters to have only one vertex to test at run-time. We show that by computing point buffers inside leaf nodes of depth 12 and applying our rules described below, our algorithm is already I/O bound.

### Point buffer creation and point in N-polygon inclusion query

At run-time, when testing the polygon inclusion status for point queries lying inside a node with geometry, it is not optimal to carry the same

procedure of determining the polygon inclusion of a group, following a ray to the outside of the quadtree each time. One could stop counting intersections when the ray reached the bounding box limits of the polygon of interest, however this can involve traversing several triangles with and without geometry. A faster strategy is to stop the ray when the ray enters a stable node (void of geometry) and apply the following additional rules (3,4,5,6,7) to infer the polygon inclusion status (please refer to Fig. 10 bottom):

**Rule 3** – Given an even or zero number of intersections for a polygon ID and the stable node entered is classified as being outside that polygon, the point query is deemed to be outside that polygon (rays starting from red points in node B, reach the outside stable node C).

**Rule 4** – Given an odd number of intersections for a polygon ID and the stable node entered is classified as being outside that polygon, the point query is deemed to be inside that polygon (rays starting from blue points in node B, reach the outside stable node C).

**Rule 5** – Given an even or zero number of intersections for a polygon ID and the stable node entered is classified as being inside that polygon, the point query is deemed to be inside that polygon (for illustration purposes, rays fired towards the left and starting from blue points in node B, test geometry in B and traverse 3 subnodes of the same size with geometry to reach a large inside stable node to the left of node E).

**Rule 6** – Given an odd number of intersections for a polygon ID and the stable node entered is classified as being in that polygon, the point query is deemed to be outside that polygon (rays starting from red points in node B, reach a large inside stable node to the left of node E).

We have so far considered polylines whose vertex spacing is similar to the resolution of the tree. During the landmass extraction process, our horizontal rays were guaranteed to traverse the quadtree to a point outside it, therefore intersecting any polygons in its path. For shorter rays in our point queries, we need to cater for polylines with a vertex spacing that is greater than the edge length of the leaf nodes of the tree (Fig.10-top).

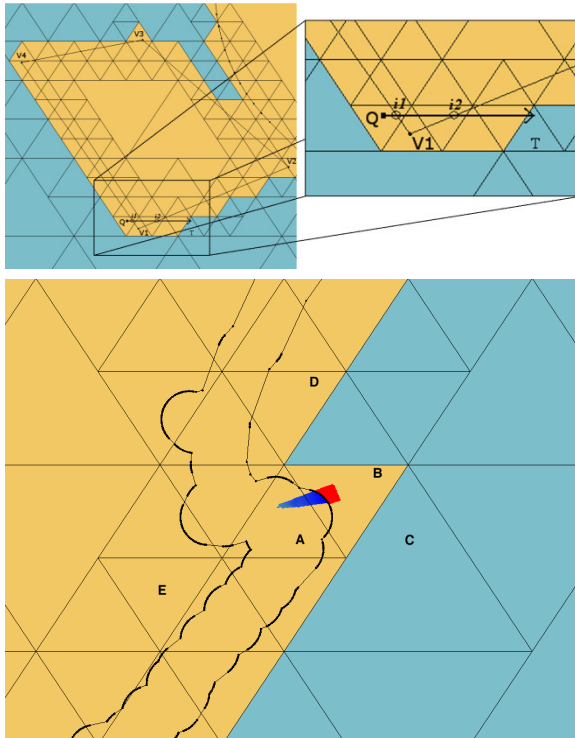


Figure 10. Top: Rule 7. Bottom: Rules 3-7.

Specifically the ray test retrieves the edge information of a node flagged with geometry or containing a vertex, and tests the edge for intersection. Since the ray is no longer guaranteed to traverse to the outside of the tree, the intersection point with the edge is tested to see if it is inside the triangle with the ray segment being tested. If the intersection is outside, it is not counted, and is instead only counted in the triangle where the ray intersects the edge. This then allows one to use the following rule to determine the polygon inclusion set of the query:

**Rule 7** – the polygon inclusion set of a pre-calculated node whose ray traversed to the outside of the quadtree, can be re-used in the counting of any horizontal ray test that shares its original ray path.

Still, it is not optimal to cast a ray from a point query towards a node void of geometry, as adjacent triangles still need to be retrieved along the ray path. Instead we pre-compute the multiple polygon inclusion of the center position (point buffer) of

every leaf triangle with geometry. At run-time point queries inside leaf nodes with geometry simply cast a ray from their position towards their triangle center position and rules 3-7 are applied to infer the polygon inclusion status. When pre-computing the inclusion status of a point buffer we apply rule 8 to allow the ray to follow the shortest path (Fig.9-right) to a stable node with known inclusion status and apply rule 9. We note that the presented rules for rays have similarities with the process of finding the topological genus of an object in that they require the object to be bended in space to test for equivalence.

**Rule 8** – the polygon inclusion set of a pre-calculated node (buffer status) void of geometry whose ray traversed to the outside of the quadtree, can be re-used in the counting of any ray from any direction, in other words ray segments can change direction, and the counting of local intersections enclosed in each traversed triangle that the buffer status can still be used in the end to infer inclusion for the ray test as a whole.

**Rule 9** – In the limit of the subdivision, nodes void of geometry are point buffers with surrounding triangles void of geometry.

Figure 9-right shows the smallest number of triangles that need to be traversed from that triangle to find an adjacent triangle void of geometry. These numbers are computed as follows, we create an array only with pointers to leaf nodes with geometry, and access each node (A) in the array and retrieve the triangle's three adjacent neighbors, if one of them is void of geometry (B) then we mark A with the distance 1, and we record in A which of its three edges leads to B (this is the ID from 1-3 that is used for retrieving the adjacent triangle that is the shortest ray path later in the point-buffer status calculation), if all neighbors had geometry we leave A untouched, we do the same with the remaining nodes in the array. We then increment the distance to 2 and look in the array to find any unmarked node (A) that has an adjacent neighbor (B) with a marked distance of 1, if there is we mark A with a distance of 2, hence working backwards and inwards. The process ends when there was no change made when iterating through the array.

## 4. RESULTS

Timings for the construction of the various phases of Trixel Buffers are given in Table 2. Memory statistics are given in Table 3. We tested Trixel Buffers with 3 different bathymetric datasets of different sizes (Table4). All the results in our article were carried out on a 2.5Ghz Duocore machine with 2GBytes of RAM, except if indicated otherwise. Figure 1, shows the Point in N-polygons



level	Subdivision	Subdivision/ leaf size refinement	Polygon line extraction	Refinement/ zero filling	Automatic Interior/ Exterior extraction	Calculate minimum distance 2 void node for point buffers	Create point buffers	Total time
1	<1s	<1s	<1s	<1s	<1s	<1s	<1s	1s
2	<1s	1s	<1s	<1s	<1s	<1s	<1s	2s
3	<1s	1s	1s	<1s	<1s	<1s	<1s	2s
4	1s	1s	1s	<1s	<1s	<1s	<1s	3s
5	1s	2s	1s	<1s	<1s	<1s	<1s	4s
6	1s	2s	1s	<1s	<1s	<1s	<1s	4s
7	2s	1s	2s	<1s	<1s	<1s	1s/	5s
8	2s	2s	1s	<1s	<1s	<1s	1s	6s
9	1s	2s	2s	<1s	<1s	<1s	1s	6s
10	2s	1s	2s	<1s	<1s	<1s	1s	7s
11	2s	2s	2s	<1s	2s	<1s	2s	10s
12	2s	2s	3s	1s	6s	2s	2s	17s
13	3s	2s	3s	1s	12s	3s	6s	30s
14	3s	2s	5s	1s	29s	5s	12s	58s
15	3s	2s	8s	1s	1m07s	12s	24s	1m58s
16	3s	2s	15s	1s	2m26s	26s	48s	4m04s
17	3s	2s	28s	3s	5m24s	55s	1m37s	8m40s

Table 2. Trixel Buffers construction timings.

inclusion queries results using a combined polygon set of 270 000 vertices (convex and concave with av. spacing of 136 meters). left: result of 1 757 billion bathymetric point queries (Solent) in an area of 9.7x6.8 km, the point in N polygons inclusion test took 1h 36 min (I/O bound) with a new spatial tree depth 12 (leaf node size 1600 meters) rather than depth 16 (100 meters) for equivalent performance of competing methods and assuming only convex polygons in the later; center: 61 million bathymetric points (Kirkwall) in an area of

Sub-division Level	# non-leaves	# leaves	# leaves with geometry	# leads without geometry	Node width corresponding geodetic (Km)	size RAM (MB)
0	0	1	1	0	6026.067	<1
1	1	4	2	2	3225.863	<1
2	3	10	6	4	1637.735	<1
3	9	28	14	14	821.909	<1
4	23	70	33	37	411.333	<1
5	56	169	83	86	205.713	<1
6	139	418	212	206	102.862	<1
7	351	1,054	520	534	51.432	<1
8	871	2,614	1,323	1,291	25.716	<1
9	2,194	6,583	3,415	3,168	12.858	1
10	5,609	16,828	8,222	8,606	6.429	2.4
11	13,831	41,494	20,143	21,351	3.214	6
12	33,974	101,923	48,327	53,596	1.607	14
13	82,301	246,904	100,744	146,16	0.803	35
14	183,045	549,136	206,843	342,293	0.401	79
15	389,888	1,169,665	419,403	750,262	0.200	169
16	809,291	2,427,874	844,367	1,583,507	0.100	352
17	1,653,658	4,960,975	1,694,424	3,266,551	0.050	721

Table 3. Trixel Buffers memory consumption.

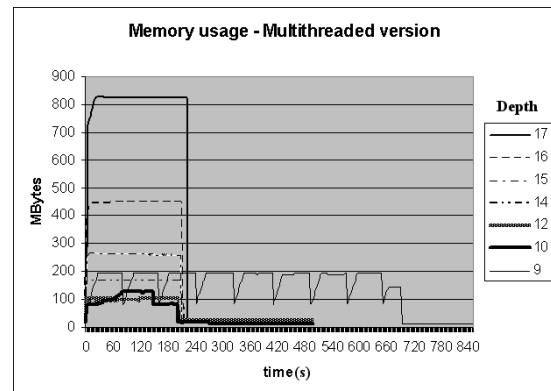


Figure 12. MT(read&amp;queries)performance using a fixed 120MB read buffer and different depth tree.

19x16 km, the point in N polygons inclusion test took ~3.36 mins with the same tree; right: the root of the quadtree in light blue project is one of

example1 - 170315pts (4.7KB) valid/inv. 104391v 65924i					Kirkwall - 61068103pts (1.8GB) 23873984v 37194119i					Solent - 1757890806pts (54GB) 669641232v 1088249574i				
	Time (MT)	Time (ST)	buff.	unbuff.	Time (MT)	Time (ST)	buff.	unbuff.		Time (MT)	Time (ST)	buff.	unbuff.	
read	1s	1s			3m34s	8m29s				1h35m27s	3h53m57s			
level														
1	14m32		0	170315			0	0				0	0	
2	14m25		0	170315			0	0				0	0	
3	12m13	54m8	0	170315			0	0				0	0	
4	1m10	3m18	0	170315			0	0				0	0	
5	24s	1m03	0	170315			0	0				0	0	
6	20s	53s	0	170315			0	0				0	0	
7	4s	12s	0	170315	2h24m		0	61068103				0	0	
8	2s	8s	0	170315	47m52		0	61068103	14h7m18			0	1757890806	
9	2s	6s	0	170315	12m11		0	61068103	4h14m07			0	1757890806	
10	1s	4s	0	170315	3m36		247663	60820440	1h46m26			291548	1757599258	
11	<1s	2s	0	170315	3m37		8591533	52476570	1h44m11			291548	1757599258	
12	<1s	1s	0	170315	3m37		25101795	35966308	1h36m42			129667487	1628223319	
13	<1s	1s	69190	101125	3m37		40185465	20882638	1h36m47			966674193	791216613	
14	<1s	1s	88307	82008	3m37		49803339	11264764	1h36m47			1331590546	426300260	
15	<1s	1s	137395	32920	3m36	8m39	55228057	5840046	1h36m39	3h56m57		1527122289	230768517	
16	<1s	1s	155748	14567	3m36	8m37	58098312	2969791	1h36m33	3h55m05		1634977067	122913739	
17	<1s	1s	162400	7915	3m36	8m40	59571359	1496744	1h36m30	3h54m56		1695689619	62201187	

Table 4. Trixel Buffer query performance.

potentially 20 icosahedron triangles covering the Earth, the darker blue polygon/the project limit polygon spans 1595x962km, the centre image [A] and the image on the left [B] cover areas smaller than the displayed characters A, B. We believe that when using 20 root triangular quadtrees, exiting horizontal rays could test for intersection the edges of the remaining 19 triangles to determine a tracing point for the ray in an adjacent root quadtree. In an earlier version of our system, before we introduced the “shortest triangle path information” or “point buffers at run-time”, we casted a horizontal ray from the point query to a stable node void of geometry to infer the polygon status. This algorithm proved to be twice as fast as strategies that trace a horizontal ray out of the spatial tree using a tree of depth 16, and still significantly faster than a ray strategy that stops a ray at the polygon limits of the polygons in question. The automatic interior/exterior extraction of a tree of depth 17 used to take 49 minutes to compute with a 3GHz processor rather than 5 minutes (Table 2) because the adjacent triangles were grouped into several individual horizontal triangle strips of the same height, rather than just 1 test made and copied for the whole group. Point validity results were inspected visually and matched the visual results of the [Tay94a] algorithm that does not use spatial databases, this algorithm which uses a standard ray test took over 4 days to process the Kirkwall data set on the 3GHz machine. Our algorithm later became I/O bound using point buffers, with multithreading/MT (tree of level 12) and also with a single thread/ST (tree of level 15). Timings for just reading point queries from disk are given in the row labeled read in Table 4. For the multithreaded version we read blocks of 1000 points and inserted them into the back of a queue, blocks in the front of the queue were removed for processing while reading, if the queue reached its full capacity (5000 blocks, 120Mb) reading was stopped until all blocks in the queue were processed, this did not happen in the I/O bound cases (Fig.12).

## 5. DISCUSSION

Although the performance of our Point-in N polygon inclusion tests is I/O bound, it takes over a minute to render a binary visualization file of results (Fig.1, center), it would be nice to organize the file spatially during the queries so as to enable frustum culling&sampling according to zoom level for interactive rendering.

## 6. CONCLUSIONS

We developed Trixel Buffers and nine rules that extend the Gauss-Jordan theorem to be used efficiently with hierarchical spatial databases. The concept can be used for determining the inclusion in 3D/octrees, and extend other quadtree methods.

## 7. ACKNOWLEDGMENTS

The authors wish to thank the United Kingdom's Hydrographic Office for providing the Bathymetric data. Funding by UKHO CONTRACT No HA294/024/009.

## 8. REFERENCES

- [Berg97a] Berg, M., Kreveld, M., Overmars, M. and Schwarzkopf, O. Computational Geometry Algorithms and Applications. Springer-Verlag.
- [Fek90a] Fekete, G. Rendering and managing spherical data with Sphere Quadtrees. Proc. of IEEE Visualization, 14:176-186, 1990.
- [Hai94a] Haines, E. Point in Polygon Strategies. Graphics Gems IV, pp.24-46, 1994.
- [Edel86a] Edelsbrunner, H., Guibas, L. J., and Stolfi, J. Optimal point location in a monotone subdivision. SIAM J. Comput, 1986.
- [Ili06a] Iliffe, J., Ziebart, M., Turner, J., Oliveira, J. F. and Adams, R. The VORF project – Joining up Land and Marine Data. GIS Professional, Issue 13, November/December, 2006, pp.24-26.
- [Kir83a] Kirkpatrick, D. G. Optimal search in planar subdivisions. SIAM J. Comput, 1983.
- [Oli06a] Oliveira, J. F., and Buxton, B. F. PNORMS: Platonic derived Normals for error bound compression. ACM VRST, 2006.
- [Pat06a] Patrascu, M., Planar Point Location in Sublogarithmic Time. Proceedings of the 47<sup>th</sup> Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 325-332, 2006.
- [Pov04a] Poveda, J., Gould M., and Oliveira, A. A New Quick Point Location Algorithm. Springer's Lecture Notes in Comp.Sci.3289.
- [Prep85a] Preparata, F. P., and Shamos, M. I. Computational Geometry an Introduction. Springer-Verlag,, ISBN 0-387-96131-3, 1985.
- [Ran02a] Randall, D. A., Ringler, T. D., Heikes, R. P., Jones, P., and Baumgardner, J. Climate Modeling With Spherical Geodesic Grids, Computing in Science & Engineering, 2002.
- [Sam90a] Samet, H. Applications of Spatial Data Structures. Springer-Verlag, Addison Wesley.
- [Sar86a] Sarnak, N., Tarjan, R. E. Planar Point Location Using Persistent Search Trees. Communications of the ACM, 29(7), 1986.
- [Ham13a] Hamilton, C.J. Views of the Solar System: <http://www.solarviews.com>
- [Tay94a] Taylor, G. E. Point in polygon test. Survey Review, 32(254): 479-484, 1994.
- [Xu03a] Xu, Z. S., and Tang, L. An efficient rejection test for ray/triangle mesh intersection. Journal of Software, 14(10): 1787-1795, 2003.



## 2.5D Clip-Surfaces for Technical Visualization

Matthias Trapp      Jürgen Döllner

Hasso-Plattner-Institut, University of Potsdam, Germany

{matthias.trapp | juergen.doellner}@hpi.uni-potsdam.de

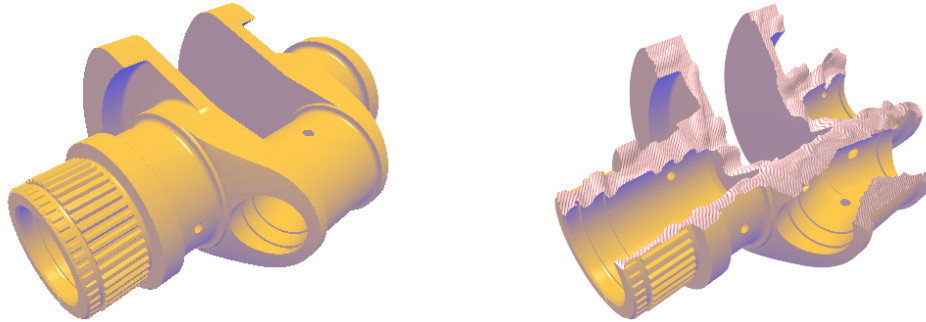


Figure 1: Application of a 2.5D clip-surface to a virtual 3D model of a crank.

### ABSTRACT

The concept of clipping planes is well known in computer graphics and can be used to create cut-away views. But clipping against just analytical defined planes is not always suitable for communicating every aspect of such visualization. For example, in hand-drawn technical illustrations, artists tend to communicate the difference between a cut and a model feature by using non-regular, sketchy cut lines instead of straight ones. To enable this functionality in computer graphics, this paper presents a technique for applying 2.5D clip-surfaces in real-time. Therefore, the clip plane equation is extended with an additional offset map, which can be represented by a texture map that contains height values. Clipping is then performed by varying the clip plane equation with respect to such an offset map. Further, a capping technique is proposed that enables the rendering of caps onto the clipped area to convey the impression of solid material. It avoids a re-meshing of a solid polygonal mesh after clipping is performed. Our approach is pixel precise, applicable in real-time, and takes fully advantage of graphics accelerators.

**Keywords:** clipping planes, real-time rendering, technical 3D visualization

### 1 INTRODUCTION

Clipping planes are often used in illustrative technical visualization for the exploration of complex 3D shapes. Resolving spatial occlusion, they enable a user to view the internal or hidden parts and assembly processes of computer-aided design drawings more efficiently by identifying any intersection that conflicts within the assembly or the assembly configurations. This clipping mechanism is a fast and easily understandable technique with simple interaction [2]. There are a number of clipping variants, such as clipping into half-spaces, cross sections, exploded or cutaway-views, which have proven successful in various areas of visualization.

However, traditional planar clipping planes might not always be suitable to communicate every aspect of an illustrative technical visualization. For example, in hand-drawn technical illustrations, artists tend to communicate the difference between a cut and a model feature by using non-regular, sketchy or ragged cut lines instead of straight ones [23]. To enable the rendering of such effect, this paper presents an extension of the planar clipping planes that is denoted as *2.5D clip-surface*

(CS) (Figure 1). It enables clipping against a number of non-planar, more precisely 2.5D clipping planes, and the flexible stylization of the cutting regions. The presented approach is suitable to extend existing rendering techniques for interactive cut-away views or cross-section illustrations.

The basic idea of a 2.5D CS is simple: consider the plane equation for a traditional clipping plane  $CP = (N_x, N_y, N_z, D)$ . Using this parametric plane equation, one can decide for every given position  $P = (x, y, z) \in \mathbb{R}^3$  in which half space it is located. Clipping can then be formulated as follows:

$$\chi(CP, P) = N_x \cdot x + N_y \cdot y + N_z \cdot z + D > 0$$

This formulation can be extended by varying the distance  $D$  to the plane origin for each point  $P$  by using an additional height value  $h = f(CP, P, OM)$  that is sampled from a offset texture map ( $OM$ ) using  $f$ . This results in a new parameterization:

$$\chi(CS, P) = N_x \cdot x + N_y \cdot y + N_z \cdot z + D + h > 0$$

By introducing local distance variations, 2.5D clip-surfaces can be used to interactively create non-regular

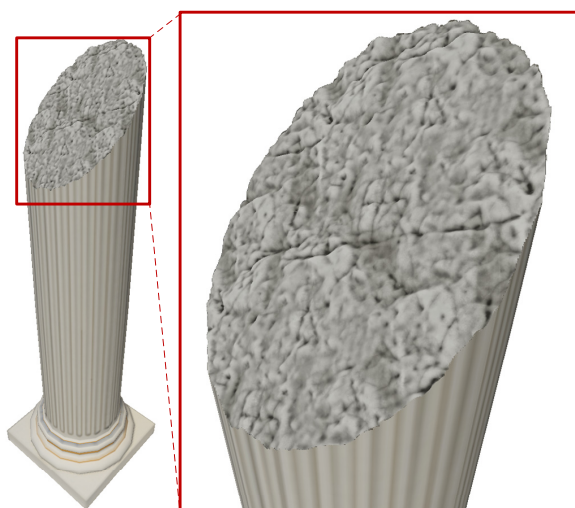


Figure 2: Application of a 2.5D clip-surface and with a cap surface to a column dataset.

clipping silhouettes while a cap-surface conveys complex inner structure of a clipped mesh (Figure 2).

While the application of CS can be implemented straight forward using fragment shader programs, the task of conveying the impression of solid material by using a *cap surface* to visualize the inner structure of an input mesh, is a challenging one. Due to the non-planarity of the clip surface, capping techniques based on stencil-buffer capabilities [3] cannot be applied to non-convex shapes, because the association of a cap surface to a clipped area cannot be decided in image-space using stencil masks.

In this work we present a concept and rendering technique that addresses the above challenges with respect to real-time, raster-based image synthesis. To summarize, this work makes the following contributions:

1. It presents a fully-hardware accelerated rendering technique that enables the application of multiple 2.5D clip-surfaces within a single rendering pass.
2. It presents a novel concept and fully hardware accelerated implementation for generating and rendering of cap surfaces.

The remainder of this paper is structured as follows: Section 2 discusses related and previous work. Section 3 introduces a rendering pipeline that implements 2.5D clip-surfaces with cap surfaces. Section 4 describes the parametrization and hardware-accelerated real-time implementation of clip-surfaces. Section 5 describes the generation and rendering of cap-surfaces. Section 6 presents application examples, discusses problems and limitations, and presents ideas for future work. Finally, Section 7 concludes this paper.

## 2 RELATED WORK

Despite the fundamentals of line and polygon clipping [9, 11] for polygonal 3D scene representations, related work mostly comprises the generation of *cut-away views* and *exploded view diagrams*. These interactive visualization techniques reveal the interior of complex 3D models by clipping either occluding parts or outer layers [19]. Traditionally, these depictions are static and often hand-crafted, thus the view point and the displayed cuts are fixed. Interactive cut-away views overcome these drawbacks by enabling the user to choose desired cut planes and cut volumes, while exploring and navigating the 3D virtual environment simultaneously.

In 1993, Lorensen presents an approach for image-based clipping using Boolean textures [20], a texture mapping technique, which is based on implicit functions to generate texture maps that are used to perform clipping during a renderer's scan conversion step. However, implicit functions are usually hard to model. Coffin et al. present a technique that enables a user to look beyond occluding objects in arbitrary 3D graphics scenes by interactively cutting holes into the occluding geometry [7]. The interactive image synthesis for this kind of virtual x-ray vision is performed using constructive solid geometry (CSG). Also using image-based CSG for rendering [14], a sophisticated approach for generating 3D cut-away views was introduced by Li et al. [19]. This work was later extended to generate interactive explosion diagrams [18]. An application of this rendering technique to mathematical surfaces [13] uses planar clipping planes. All of the previous approaches do not directly enable the stylization of the cut surfaces and the rendering performance depends on the depth complexity of the virtual scene with respect to the virtual camera [14].

A more effective image-based approach for rendering cut-aways views for geometrical complex 3D scenes is presented by Burns et al. [5]. Based on distance transformations of the depth buffer content, view-dependent cut-aways can be generated for a number of 3D objects. This approach depends on the virtual camera and always exposes the complete objects-of-interest in the context of surrounding objects. Our approach enables the creation of consistent cut-away illustration for varying virtual cameras. In [15], a system for creating illustrative cutaway renderings is presented that rely on sketch-based interfaces and stylized rendering techniques for the study of elaborate 3D models. With respect to interaction, Clifton & Pang present extensions to the traditional cutting plane for virtual reality devices. Using their hands, users interact directly with the data to generate arbitrarily oriented planar surfaces, linear surfaces, and curved surfaces [6].

In the field of volume graphics and rendering, Weiskopf et al. propose clipping methods that are

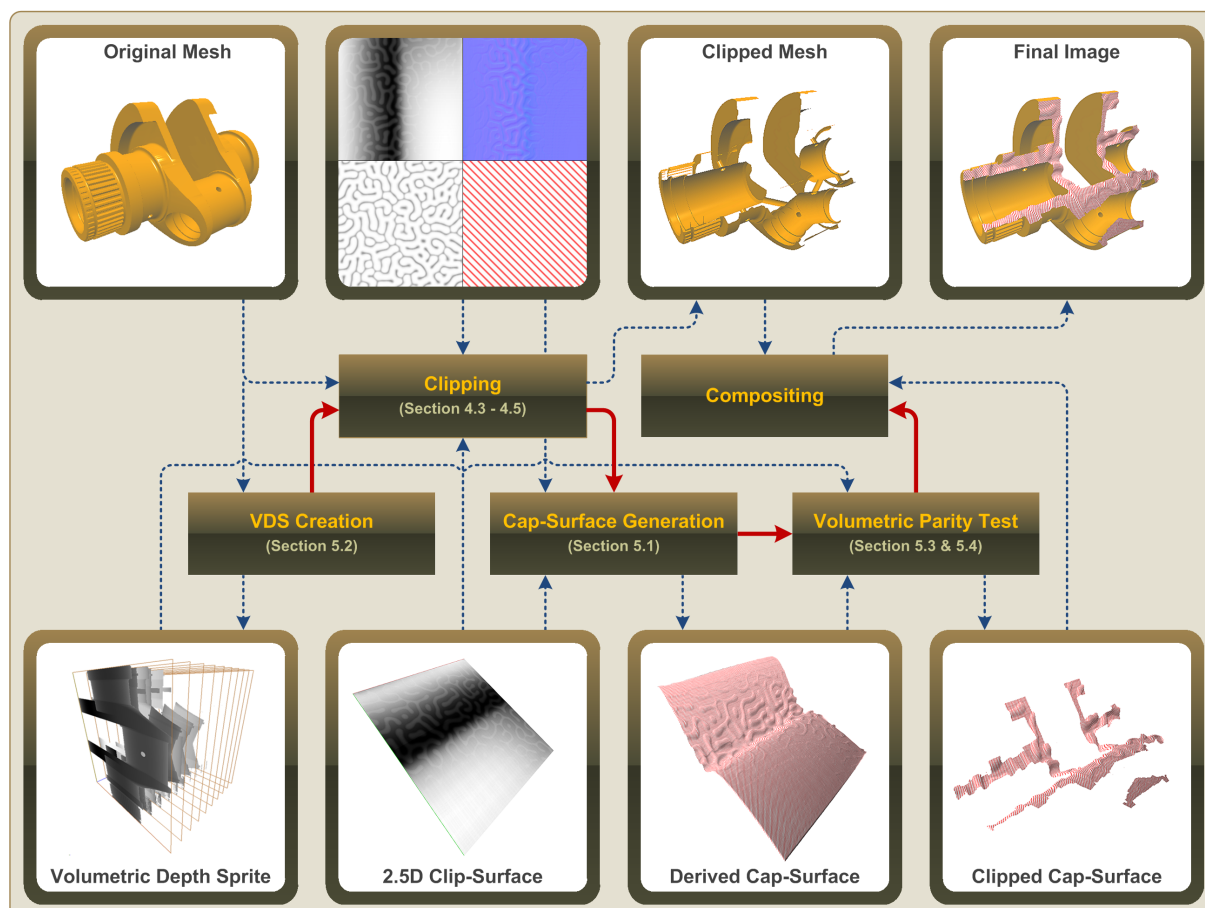


Figure 3: Conceptual overview of the rendering pipeline for 2.5D clip-surfaces and non-planar cap surfaces.

capable of using complex geometries for depth-based volume clipping [32]. It is based on an approach for volume clipping based on per-fragment operations on voxelized clip object that are used to identify the clipped regions [31]. Based on over-sampling fragment shader programs, Röttger et al. increase the rendering quality for volumetric clipping [27]. Further, Qi & Martens investigate the aspects of positioning a clipping plane within volume-rendered data. They propose three different interface prototypes that combine aspects of 2D graphical user interfaces with 3D tangible interaction devices based on wireless vision-based tracking [26]. Birkeland et al. presents a feature and context preserving clipping approach called membrane clipping [2]. These non-planar clipping planes implement selective feature preservation using an elastic membrane. With respect to the cut-away approaches described above, our contribution presents a complementing approach for creating cut-away illustration.

### 3 CONCEPTUAL OVERVIEW

The proposed image-based approach works for an arbitrary solid input mesh. Figure 3 shows the rendering pipeline for 2.5D clip-surfaces and cap surfaces applied

to the domain of technical illustration. The complete rendering process comprises the following three steps that are performed on a per-frame basis:

1. Application of clip surfaces by rendering the solid mesh into the framebuffer with applied clipping (Section 4).
2. Automatic generation of a cap surface from the clip-surface. This is implemented using a polygonal cap surface that is derived from the clip-surface parametrization. GPU based-mesh refinement [4] is applied to fit the subdivision of the cap mesh to the resolution of the offset map (Section 5.1).
3. Clipping of the cap mesh by performing a *volumetric depth test* on a per-fragment basis. It determines if a fragment lies inside the volume and thus is associated with a gap, or if it is located outside the input shape and therefore is discarded. In this step per-vertex displacement mapping, and per-fragment shading, and texturing is also performed (Section 5.2 and 5.3).

Subsequently, the intermediate results of each stage (i.e., the fragments of the clipped mesh and clipped cap-surface) can be composited by rendering to the frame

buffer successively or by using an additional compositing step to apply post-processing effects (e.g., edge-enhancement [22]). Therefore, the intermediate results are rendered into off-screen buffers [28] and composited using an additional post-processing pass.

## 4 IRREGULAR CLIP-SURFACES

As described previously, a 2.5D clip-surface is an extension of the standard clipping plane by offsetting each point on the plane using height variances. Instead of modeling these local height variances using implicit functions [20] they are represented using a texture map that contains height values.

Given a surface parametrization (Section 4.1) including a respective offset map (Section 4.2), pixel-precise clipping (Section 4.3) can be performed for every fragment position during rasterization using a fragment shader program (Section 4.4). This approach also enables the rendering of multiple clip-surfaces within a single rendering pass (Section 4.5).

### 4.1 Parameterization

Briefly, a 2.5D clip-surface  $CS = (O, U, V, S, OM)$  can be modeled using the following five parameters (cf. Fig. 4): an origin  $O \in \mathbb{R}^3$ , two orthogonal direction vectors  $U, V \in \mathbb{R}^3$  (to support anisotropic adjustments), a scaling vector  $S = (S_x, S_y, S_z) \in \mathbb{R}^3$ , and an offset texture map  $OM$ . The normal vector  $N$  of the plane is implicitly given by  $N = \|U\| \times \|V\|$ . A CS can be extended with further attributes (e.g., normal, diffuse, and light texture maps) that define the appearance of the cap surface.

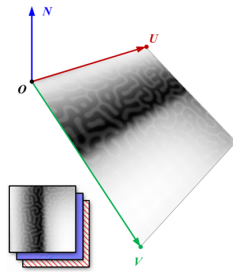


Figure 4: Parameters of a 2.5D clip-surface.

### 4.2 Offset Maps

The top row of the right Figure 5 shows some examples of 2D offset maps. Basically, it is an image-based representation of a 2.5D clip surface that is used for clipping. Offset maps can be easily combined (e.g., Column 3), using standard image blending operations. The accompanying normal maps (Row 2) and occlusion maps (Row 3) can be computed based on the offset maps using a preprocessing step at run-time.

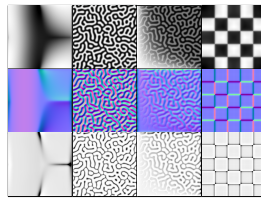


Figure 5: Examples of different offset, normal, and occlusion maps (top to bottom).

Representing the irregular clip surface using texture maps offers a number of advantages: (1) it enables a high design freedom since they can be created using standard imaging software; (2) they can be easily exchanged and combined; and (3) can be easily represented and modified on GPU.

### 4.3 Pixel-precise Clipping

Given an arbitrary shaped solid mesh and a CS, clipping is performed at fragment level. For each fragment with a clip-space coordinate  $P$  the following Boolean function:

$$\chi(CS, P) = \begin{cases} 1 & P \bullet N - N \bullet O + f(OM, T) \cdot S_z < 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$T = \left( \frac{(P_N - O) \bullet \hat{U}}{\|U\|} \cdot S_x, \frac{P_N - O \bullet \hat{V}}{\|V\|} \cdot S_y \right) \quad (2)$$

$$P_N = P - ((P - O) \bullet N) \cdot N \quad (3)$$

is evaluated. Here, the function  $f$  delivers a scalar value  $h \in [0, 1]$  by first, generating texture coordinates  $T$  using Equation 2 and 3, and second, sampling the offset map  $OM$ , and finally scaling the resulting height sample by  $S_z$ . The sampling step depends on the type of offset map used, i.e., it differs for 1D, 2D, or 3D representations for an offset map. A point  $P$  is clipped if the half-space test in Equation 1 resolves to 1 (*true*). This function can be efficiently implemented using a fragment shader program and the fragment discard functionality.

### 4.4 Fragment Shader Implementation

Listing 1 shows an OpenGL shading language (GLSL) implementation for the clip-surface parameterization. It first assembles the plane equation from the parameterization, then project the input point onto that plane and compute the required texture coordinates. Using these, the offset texture map is sampled and half-space test is performed. This function can be executed for a number of surfaces within a single rendering pass. Prior to shader execution, the parameterization is encoded in a matrix representation.

```
bool clipSurface(
    const in mat4 config,
    const in vec4 P,
    const in sampler2D offsetMap) {
    // compute plane parametrization in eye space...
    vec3 O = (gl_ModelViewMatrix *
              vec4(config[0].xyz, 1.0)).xyz;
    vec3 A = gl_NormalMatrix * normalize(config[1].xyz);
    vec3 B = gl_NormalMatrix * normalize(config[2].xyz);
    vec3 N = cross(A, B);
    // project fragment coordinates onto plane
    vec3 PN = P.xyz - dot(P.xyz - O, N) * N;
    // compute clip texture coordinates...
    float s = dot(PN - O, A) / length(config[1].xyz);
    float t = dot(PN - O, B) / length(config[2].xyz);
    // fetch height...
    float height = texture2D(offsetMap,
                             vec2(s, t) * config[3].st).x;
    // compute reference plane...
    float plane = dot(point.xyz, N) -
                  dot(N, O) + (height * config[3].z);
    // perform clipping if surface is enabled...
    return (plane < 0.0 && bool(config[3].w));
}
```

Listing 1: GLSL implementation to evaluate 2.5D clip-surface for a given point.



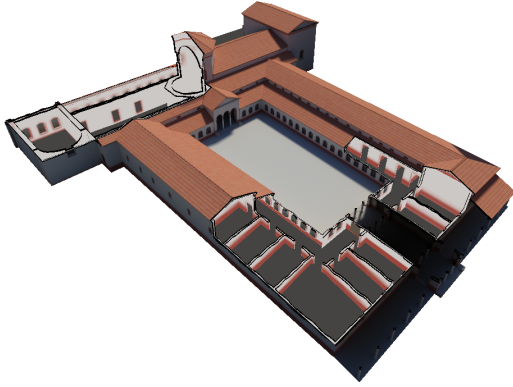


Figure 6: Rendering multiple cut-away views of a virtual 3D building model by applying two 2.5D clip-surfaces.

#### 4.5 Multiple Clip-Surfaces

To enable multiple cut-away views, a number of clip-surfaces are evaluated within a single rendering pass. Figure 6 shows an example for applying two different clip-surfaces to reveal parts of the building footprint and internal structures, such as walls and doorways, which otherwise would be hidden to the viewer.

The quality of the 3D models (in terms of modeled interior, solid walls, and consistency of polygon orientation), is important for the resulting visual impression. Besides additional configuration issues, the application of cut-away views introduces a number of challenges and technical implications to the visualization framework: for example, Figure 6 shows shading and shadow discontinuities for areas inside and outside the building. These are caused by using a pre-computed lighting approach, which is only valid for views from outside the building. This effect can be compensated partially by using image-based global lighting approaches that approximate ambient occlusion [21].

### 5 CAP SURFACES

For certain types of renderings, it can be desired to show the inner material properties after applying clipping. The sections marked in red in Figure 7 represent the surface of the inner material of a broken crank denoted as *caps*. For planar clipping planes, capping is usually performed by a stencil-buffer technique using the back-face and front-face polygon orientation information [3]: a stencil mask representing the visible back-faces determine the areas where to render the cap. Since the non-regularity of the clip surface, this approach is not suitable to be used for our purposes.



Figure 7: Surfaces of a broken real-world crank marked in red.

This paper presents an image-based approach for the real-time rendering of cap surface for a 2.5D clip-surface. It is suitable for any solid input mesh, i.e., a mesh that is modeled "water-tight" in real-time.

#### 5.1 Rendering of Cap Surfaces

Conceptually, the rendering of cap surfaces comprises two main steps: the *generation* of a cap surface and the *clipping* of all surface parts that are *outside* the volume enclosed by the input mesh.

During the *cap-surface generation step*, a triangulated quad is generated using the  $U$  and  $V$  parameters, which resembles the 2.5D cap surface (cf. to *Derived Cap-Surface* in Fig. 3). This mesh is then adaptively refined on GPU [4] to achieve a sufficient vertex density. Here, a generic refinement pattern (stored at full resolution as a vertex buffer on the GPU memory) is used to virtually create additional inner vertices for the generated cap-surface. Subsequently, each vertex of the refined mesh is displaced using the offset map  $OM$  as well as textured and shaded using specific diffuse, normal, and occlusion maps. The generation of texture coordinates is similar as for the clipping approach (Eqn. 1 to 3).

For the *cap-surface clipping step* during the rasterization of the refined cap surface, each fragment is tested if it lies inside or outside the volume enclosed by the input mesh in order to determine where the gaps are located that have to be covered by the surface. Fragments that fail a *volumetric parity test* (Section 5.3), which is based on an *image-based volumetric representation* of the input mesh (Section 5.2), are discarded using a fragment shader program (cf. to *Clipped Cap-Surface* in Fig. 3 and Section 5.4 for implementation).

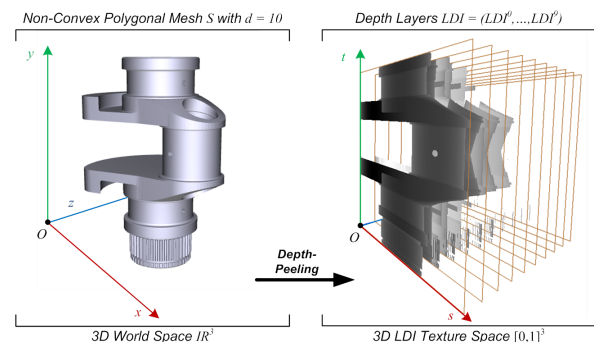


Figure 8: Example of an volumetric depth sprite. The non-convex polygonal mesh  $S$  is depth-peeled into layers of unique depth complexity.

#### 5.2 Volumetric Depth Sprites

A *volumetric depth sprite* (VDS) is an image-based representation of a shapes volume by storing its depth values along parallel viewing rays [30]. A VDS extends the concept of LDIs [29] that contain layers of unique depth complexity. An LDI is a view of the scene from a single input camera view, but with multiple pixels

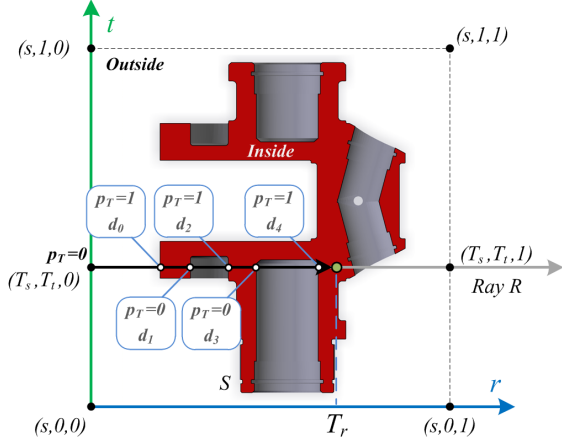


Figure 9: Ray marching through an LDI representation of the complex input shape shown in Figure 8. A ray  $R$  intersects the depth layers  $LDI^i$  at five points and adjusts the rays parity  $p_T$  accordingly.

along each line of sight. The size of this representation grows linearly with the observed depth complexity in the scene. Figure 8 shows an example of a VDS derived from a complex 3D shape.

A VDS representation consists of the following components:  $VDS = (G, LDI, d, w_i, h_i)$ . Here,  $G \in \mathbb{R}^3$  denotes the reference position of the VDS in world-space coordinates. The depth complexity of  $S$  is denoted as  $d \in \mathbb{N}/\{0,1\}$ . The layered depth image consist of  $d$  depth maps  $LDI = (LDI^0, \dots, LDI^{d-1})$ . The initial texture resolution of width and height is given by  $w_i, h_i \in \mathbb{N}$ . To obtain a depth value  $d_i \in [0, 1] \subset \mathbb{R}, 0 \leq i \leq d-1$  in the  $i^{th}$ -depth layer for a 2D point  $(s, t) \in [0, w_i] \times [0, h_i]$ , the 3D texture is sampled in LDI texture space using the coordinates  $LDI_{(s,t)}^i = (s, t, i)$ .

The creation of a VDS is performed within a preprocessing step using multi-pass render-to-texture in combination with depth-peeling [8]. Given a solid polygonal mesh  $S$ , the associated  $LDI$  can be generated by performing the following three steps:

1. The shape is scaled uniformly to fit into the unit volume  $[0, 1]^3$ . A camera orientation and an orthogonal projection is chosen that covers this unit volume with the near and far clipping planes adjusted accordingly.
2. The depth complexity  $d$  is computed and a 3D texture or 2D texture array is created with an initial resolution of width  $w_i$ , height  $h_i$ , and depth  $d$ .
3. Finally, depth-peeling [8] in combination with RTT is performed. The solid  $S$  is peeled using linearized depth values using a  $W$ -buffer [17].

### 5.3 Volumetric Parity Test

Real-time volumetric tests enable a binary partition of a given arbitrary shape on vertex, primitive, and fragment level [30]. They have a number of applications in real-time rendering and interactive visualization, such as pixel-precise clipping, collision detection, and rendering with hybrid styles [12]. Given a VDS, a *volumetric parity test* (VPT) classifies a point  $P$  with respect to its position in the shape's volume: it can either be *inside* or *outside*. To model such test, a Boolean *coordinate parity*  $p_T \in \{0, 1\}$  is used. Before testing  $P$ , it must be transformed into the specific 3D LDI texture space. For a point  $P$  in world-space coordinates, the transformed coordinate  $T = (T_s, T_t, T_r)$  can be obtained by  $T = M \cdot P$ .

The matrix  $M$  is defined by  $M := T(C) \cdot S \cdot B \cdot T(-G)$ . Where  $B$  is a rotated ortho-normal basis of the VDS.  $P$  is transformed into the LDI texture coordinate space ( $B \cdot T(-G)$ ), scaled by  $S$ , and translated ( $T(C)$ ) into the LDI origin  $C = (0.5, 0.5, 0.5)$ .

Subsequently, a ray  $R = [(T_s, T_t, 0), (T_s, T_t, 1)]$  is constructed that marches through the depth layers  $LDI^i$  and compares  $T_r$  with the stored depth values. Starting with an initial parity,  $p_T$  is swapped every time  $R$  crosses a layer of unique depth complexity (cf. Figure 9). This test can be formulated as  $p_T = VPT(T, LDI)$  with:

$$VPT(T, LDI) = \begin{cases} 1, & \exists d_i \wedge \exists d_{i+1} : d_i \leq T_r < d_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$d_i \in LDI_{(T_s, T_t)}^i \quad d_{i+1} \in LDI_{(T_s, T_t)}^{i+1}$$

### 5.4 Fragment Shader Implementation

The implementation of the ray-marching algorithm needs to iterate over the number of depth maps stored in the 3D texture, which represents the LDI. Listing 2 shows the GLSL source code that implements the VPT. The performance of this algorithm depends on the number of samples the VPT has to perform. The presented volumetric test consists of less than 20 assembler instructions per executed loop.

```
bool volumetricParityTest(
    const in vec4 T, // coordinate in LDI-space
    const in sampler3D LDI, // layered depth image
    const in int depth, // depth complexity ds
    const in bool initParity) // initial parity p
{
    // initial parity; true = outside
    bool parity = initParity;
    // compute offset to address texture slices
    float offset = 1.0 / float(depth);
    // for each texture layer do
    for(float i = 0.0; i < float(depth); i++){
        if(T.r < texture3D(LDI, // perform depth test
            vec3(T.st, offset * i)).x) {
            parity = !parity; // swap parity
        } //endif } //endfor
    return parity;
}
```

Listing 2: GLSL implementation of the VPT.



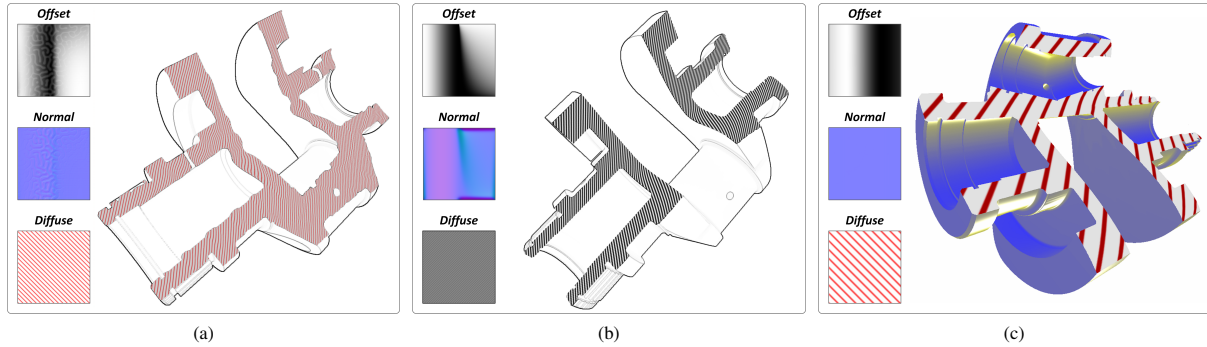


Figure 10: Examples for applying different parameterizations of a 2.5D clip-surface to a virtual 3D model of a crank. The insets show the respective 2D offset, normal, and diffuse texture maps used for stylization.

## 6 RESULTS & DISCUSSION

### 6.1 Application Examples

Figure 10 shows results for the proposed rendering techniques using different offset, normal, diffuse, and offset texture maps that enable stylization and appearance variations. For example, technical illustration styles can be achieved using hatch textures for the cap surfaces in combination with image-based edge enhancement [22] (cf. Figure 10(a) and 10(b)) or Gooch Shading [10] (cf. Figure 10(c)). Further, Figure 11 shows an application example that uses solid 3D textures [16] in combination with occlusion maps for texturing the cap surface.

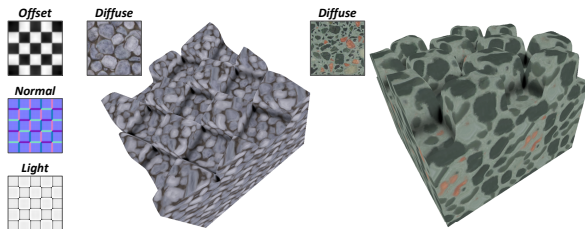


Figure 11: A single 2.5D clip-surface is applied to a cube using light maps and solid textures for the stylization of the cap-surface.

### 6.2 Rendering Performance

We tested the performance of a clip-surface with a single cap-surface using a NVIDIA GeForce GTX 285 GPU with 2048 MB video RAM on a Intel Xeon CPU with 2.33 GHz and 3 GB of main memory a viewport resolution of  $1024 \times 768$  pixels. The test model (Figure 10) comprises 50,394 vertices and 99,999 faces. It performs with 131 frames-per-second (FPS) without clipping in comparison to 125 FPS with clipping and cap-surface enabled (with a sub-division level of 256 and an LDI resolution of  $1024^2$  pixels). The applied cap-surface technique is fill-limited, i.e., the runtime performance depends on the number of fragments tested against the volumetric depth sprite.

### 6.3 Limitations & Improvements

The presented approach implies a number of technical and conceptual limitations. Despite requiring a watertight mesh, the rendering technique is limited by two drawbacks: (1) it requires an additional data structure (VDS), which is created during a preprocessing step and is only suitable for static meshes; and (2) to obtain a high visual quality, a sufficient vertex density of the cap surface is required. With respect to this, matching the tessellation factor to the screen resolution, which is required to avoid gaps between the clipped mesh and the cap surface, is an open research question.

With respect to the current implementation, there are numerous ways for future work. Instead of using volumetric depth sprites, which are generated in a preprocessing step, one could perform the volumetric parity test based on a  $k$ -Buffer implementation [1] that is generated per-frame and per-object. Additionally, this enables also the usage of dynamic objects. Furthermore, to achieve a sufficient vertex density for the mesh that represents the cap surfaces, hardware-accelerated and programmable tessellation [24] could be of interest. With respect to this, the application of relief mapping [25] to render micro structures of the cut surface could be researched.

## 7 CONCLUSIONS

This paper presents a concept and real-time rendering technique for implementing interactive 2.5D clip-surfaces. The implementation is fully hardware accelerated and includes an approach for rendering cap surfaces that are applicable to arbitrary solid input meshes. The main drawbacks of our approach are the necessary additional data structure, that consumes additional GPU memory as well as the need for highly tessellated surfaces in order to avoid rendering artifacts for the cap surfaces. However, these drawbacks can be addressed and counter-balanced in future implementations using advanced GPU capabilities.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the paper. This work was funded by the German Federal Ministry of Education and Research (BMBF), as part of the InnoProfile Transfer research group "4DnD-Vis".

## REFERENCES

- [1] Louis Bavoil, Steven P. Callahan, Aaron Lefohn, João L. D. Comba, and Cláudio T. Silva. Multi-fragment effects on the gpu using the k-buffer. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 97–104, New York, NY, USA, 2007. ACM.
- [2] Aasmund Birkeland, Stefan Bruckner, Andrea Brambilla, and Ivan Viola. Illustrative membrane clipping. *Computer Graphics Forum*, 31(3):905–914, June 2012.
- [3] David Blythe, Tom McReynold, Brad Grantham, Mark J. Kilgard, and Scott R. Nelson. Programming with OpenGL: Advanced Rendering. In A. Rockwood, editor, *SIGGRAPH Course*, New York, NY, USA, 1999. ACM Press.
- [4] Tamy Boubekeur and Christophe Schlick. Generic Mesh Refinement on GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '05, pages 99–104, New York, NY, USA, 2005. ACM Press.
- [5] Michael Burns and Adam Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Trans. Graph.*, 27(5):154:1–154:7, December 2008.
- [6] Michael Clifton and Alex Pang. Cutting planes and beyond. *Comput. Graph.*, 21(5):563–575, September 1997.
- [7] Chris Coffin and Tobias Hollerer. Interactive perspective cutaway views for general 3d scenes. In *Proceedings of the 3D User Interfaces*, 3DUI '06, pages 25–28, Washington, DC, USA, 2006. IEEE Computer Society.
- [8] Cass Everitt. Interactive Order-Independent Transparency. Technical report, NVIDIA Corporation, June 2001.
- [9] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
- [10] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 447–452, New York, NY, USA, 1998. ACM.
- [11] Günther Greiner and Kai Hormann. Efficient clipping of arbitrary polygons. *ACM Trans. Graph.*, 17(2):71–83, April 1998.
- [12] Roland Jesse and Tobias Isenberger. Use of Hybrid Rendering Styles for Presentation. In *Poster Proceedings of WSCG 2003*, pages 57–60, 2003. Short Paper.
- [13] Olga Karpenko, Wilmot Li, Niloy Mitra, and Maneesh Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, November 2010.
- [14] Florian Kirsch and Jürgen Döllner. Opencsg: A library for image-based csg rendering. In *USENIX 2005 Annual Technical Conference, FREENIX Track*, pages 129–140, 2005.
- [15] Sebastian Knödel, Martin Hachet, and Pascal Guitton. Interactive Generation and Modification of Cutaway Illustrations for Polygonal Models. 2009.
- [16] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):2:1–2:9, 2007.
- [17] Eugene Lapidous and Guofang Jiao. Optimal Depth Buffer for Low-Cost Graphics Hardware. In *HWWS '99*, pages 67–73, New York, NY, USA, 1999. ACM Press.
- [18] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph.*, 27(3):101:1–101:7, August 2008.
- [19] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive Cutaway Illustrations of Complex 3D Models. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM Press.
- [20] William E. Lorensen. Geometric clipping using boolean textures. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, pages 268–274, Washington, DC, USA, 1993. IEEE Computer Society.
- [21] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image Enhancement by Unsharp Masking the Depth Buffer. In *SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1206–1213, New York, NY, USA, 2006. ACM Press.
- [22] Marc Nienhaus and Jürgen Döllner. Edge-enhancement - an algorithm for real-time non-photorealistic rendering. *International Winter School of Computer Graphics, Journal of WSCG*, 11(2):346–353, 2003.
- [23] Marc Nienhaus, Florian Kirsch, and Jürgen Döllner. Illustrating design and spatial assembly of interactive csg. In *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, AFRIGRAPH '06, pages 91–98, New York, NY, USA, 2006. ACM.
- [24] Matthias Nießner, Charles T. Loop, Mark Meyer, and Tony DeRose. Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Trans. Graph.*, 31(1):6, 2012.
- [25] Fábio Policarpo, Manuel M. Oliveira, and João L. D. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 155–162, New York, NY, USA, 2005. ACM.
- [26] Wen Qi and Jean-Bernard Martens. Tangible user interfaces for 3d clipping plane interaction with volumetric data: a case study. In *Proceedings of the 7th international conference on Multimodal interfaces*, ICMI '05, pages 252–258, New York, NY, USA, 2005. ACM.
- [27] Stefan Roettger, Stefan Guthe, Daniel Weiskopf, Thomas Ertl, and Wolfgang Strasser. Smart hardware-accelerated volume rendering. In *Proceedings of the symposium on Data visualisation 2003*, VISSYM '03, pages 231–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [28] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, September 1990.
- [29] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered Depth Images. In *SIGGRAPH '98*, pages 231–242, New York, NY, USA, 1998. ACM Press.
- [30] Matthias Trapp and Jürgen Döllner. Real-time volumetric tests using layered depth images. In K. Mania and E. Reinhard, editors, *Eurographics 2008 Shortpaper*, pages 235–238. The Eurographics Association, 2008.
- [31] Daniel Weiskopf, Klaus Engel, and Thomas Ertl. Volume clipping via per-fragment operations in texture-based volume visualization. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 93–100, Washington, DC, USA, 2002. IEEE Computer Society.
- [32] Daniel Weiskopf, Klaus Engel, and Thomas Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):298–312, July 2003.